Compute (Bridgend) Ltd

Structured Data Editor (SDE)

SELCOPY/i Release 3.10

8 Merthyr Mawr Road, Bridgend, Wales UK CF31 3NH

Tel: +44 (1656) 65 2222
Fax: +44 (1656) 65 2227

CBL Web Site - http://www.cbl.com

This document may be downloaded from http://www.cbl.com/cblidoc.html

Contents

Documentation Notes 1

Summary of Changes 2
- First Edition (May 2012) 2

Introduction to SDE 4
- Terminology 5

Getting Started with SDE 7
- Handling Long Command Strings 7
- Opening an SDE Edit/Browse View 7
  - Display Records without a Structure 7
    - SDE Data Edit vs. CBLe Text Edit 8
  - Display Records using a single COBOL or PL/1 Copy Book 8
    - Create an SDE structure (SDO) from a single COBOL or PL/1 Copy Book 9
    - Create an SDE structure (SDO) Compiler Return Code 10
  - Display Records or Record Segments using multiple COBOL or PL/1 Copy Books 10
    - Create an SDE structure (SDO) from multiple COBOL or PL/1 Copy Books 10
    - Display Records or Record Segments using an SDO structure 13
  - Display a Subset of Records or Record Segments 13
    - Subset using FROM and/or FOR 14
    - Create a record FILTER dataset 14
    - Filter Unformatted Record Data 15
    - Filter Formatted Record Data 15
    - Subset using FILTER 17
- Changing the Record Display 17
  - Displaying Records of Specific Record Type(s) 18
  - Display Unformatted Record Data 19
  - Displaying a Record/Segment in Single Record View 19
  - Displaying Hexadecimal Representation of Record Data 20
  - Excluding Records from the Display 21
  - Displaying Selected Columns 21
  - Displaying the Record Headers 22
- Searching Data Records 23
  - Finding Record Data by Search String 23
  - Locating Records Using a Search Expression 24
  - Creating a Subset of Records for Display 24
- Editing Data 25
  - Typing New Record Data 25
  - Find and Replace Data 26
  - Inserting Records 26
  - Deleting Records 27
  - Moving Records 27
  - Copying Records 28
  - Duplicating Records 28
  - Undo and Redo Changes 28

SDE Concepts 30
- Record Structure Definition 30
  - Structure Definition Object 30
  - Record Type Object 30
  - Record Type Object Data Length 30
  - Structure Definition File 30
- Record Type Assignment 31
- Structured Data Fields 31
- SDE Window Views 32
  - Multi Record View 32
    - Field Column-Heading Lines 32
    - Prefix Area 33
    - Data Records 33
    - Shadow Lines 34
  - Single Record View 34
- SDE Data Formats 34
  - Table Format 34
  - Single Format 34
  - Character Format 35
  - Hexadecimal Format 35
  - Hexadecimal Dump Format 36
- SDE Data Types 36
- SDE Edit Techniques 39
- Changing Record Lengths 41
- SDE Current Window 41
- Default Record Type 41
- DB2 Table Display 42
  - Overview 42
  - DB2 Table Edit/Browse 42
- Segmented Records 43

Contents

SDE Concepts
- Structure Definition and Record-Type Assignment ... 44
- Specifying USE WHEN Segment Identification Criteria ... 44
- Segmented Records SDE Browse/Edit View ... 44
- Navigating Segmented Record Display ... 45
- Segmented Record Edit ... 45
  - Data Updates that affect Record Segment Mapping ... 45
  - Full Edit Functionality ... 45
  - Record Truncation ... 46
  - Full Edit Operations ... 46
- Expressions ... 47
  - Expression Terms ... 47
  - Expression Operators ... 48
    - Arithmetic Operators ... 48
    - Relational Operators ... 49
    - Logical Operators ... 49
    - Parentheses and Operator Precedence ... 49
- Function Calls ... 50
  - Subscripted Field Name Conflict ... 50
  - ABBREV() ... 50
  - BASPOS() ... 50
  - CHANGED() ... 51
  - CHANGEERROR() ... 51
  - CHANGEOK() ... 51
  - DATATYPEERROR() ... 51
  - DUPLICATEKEY() ... 52
  - EMPTYSLOT() ... 52
  - EXCLUDED() ... 52
  - FIND() ... 52
  - HASPOINT() ... 53
  - HASPREFIXCMD() ... 53
  - IDREQUIRED() ... 53
  - INSERTED() ... 53
  - KEYCHANGED() ... 53
  - LASTPOS() ... 54
  - LEFT() ... 54
  - LENGTH() ... 54
  - LENGTHERROR() ... 55
  - LOWER() ... 55
  - LRECL() ... 55
  - NOEOL() ... 55
  - POS() ... 55
  - RECORD() ... 56
  - RIGHT() ... 56
  - SAVEREQUIRED() ... 57
  - SEGPOS() ... 57
  - STRIP() ... 57
  - SUBSTR() ... 58
  - TRANSLATE() ... 58
  - TRUNCATED() ... 59
  - UPPER() ... 59
  - VALUEERROR() ... 59
  - WORD() ... 60
  - WORDS() ... 60
  - XRANGE() ... 60
- REXX Macros ... 61
  - Macro Path ... 61
  - SDE Profile (SDEPROF) Macro ... 61

SDE Menus and Popup Windows ... 62
- Library/Directory Member Selection ... 62
  - Prefix Line Commands ... 62
- SDE Browse/Edit Utilities Menu ... 62
  - Record-Type Options ... 63
  - Field Options ... 65
  - Record Information ... 65
  - Shadow-line Options ... 65
  - Window Options ... 66
  - Locate Options ... 66
- DB2 Browse/Edit Utilities Menu ... 66
  - Field Selection ... 66
  - Record Information ... 67
  - Shadow-line Options ... 67
  - Window Options ... 67

Contents

Command Line (Primary) Commands....68
Executing SDE Commands from a CBLe Text Edit Window....68
Command Reference Syntax Conventions....68
How to read the Syntax Diagrams....68
BOTTOM....69
BROWSE....70
CANCEL....76
CBLI....76
CHANGE....76
CHAR....81
CLIPBOARD....82
CMSG....82
CREATE LIST....82
CREATE STRUCTURE....83
CURSOR....95
DELETE....96
DISPLAY LIST....97
DISPLAY RECTYPES....98
DISPLAY STRUCTURE....99
DOWN....101
DROP, DDROP....102
DUPLICATE....103
EDIT....103
EDITDIALOG....111
EMSG....111
END....111
EXCLUDE....112
EXTRACT....116
FILE....116
FIND....117
FLIP....121
FORMAT....121
HELP....122
HEX....122
HEXDUMP....122
HIDE....123
IDENTIFY....124
IMMEDIATE....125
INSERT....125
LAYOUT....126
LEFT....127
LESS....128
LIST MODULES....130
LIST RPOS....130
LIST SDO....130
LIST STORAGE....131
LIST UKRS....131
LOCATE....132
MACRO....136
MAP....136
MARK....137
MORE....137
MSG....138
NEXT....138
NOMSG....139
ONLY....140
OPTIONS....143
PREVIOUS....144
QQUIT....145
QUERY....145
RCHANGE....146
RCOLOUR....146
RECLEN....147
REDO....148
REFRESH....148
REPLACE....149
RESET....150
RFIND....151
RIGHT....151
SAVE....153
SAVEAS....153
SAVESTRUCTURE....154
SDATA....154
SEGTYPE....154
SELECT....155
SET....156
SORT....156

Contents

Command Line (Primary) Commands
SYSCOMMAND, TSO, CMS....156
TEDIT....156
TOP....157
UNDO....157
UNFMT....158
UP....158
USE....160
VFMT....162
VIEW....162
WHERE....163
WINDOW....165
ZOOM....165

SET/QUERY/EXTRACT....167
ABBREVIATION - SET/QUERY/EXTRACT Option....168
ALT - SET/QUERY/EXTRACT Option....168
AUTOSAVE - SET/QUERY/EXTRACT Option....168
AUXDSNPREFIX - SET/QUERY/EXTRACT Option....169
CAPS - SET/QUERY/EXTRACT Option....170
COLATTRIBUTES - SET/QUERY/EXTRACT Option....170
COLOUR, COLOR - SET/QUERY/EXTRACT Option....172
COLWIDTH - SET/QUERY Option....174
COMPILER - SET/QUERY/EXTRACT Option....174
DESCRIPTION - SET/QUERY/EXTRACT Option....175
DRECTYPE - SET/QUERY/EXTRACT Option....176
DSN - SET/QUERY/EXTRACT Option....176
DSORG - SET/QUERY/EXTRACT Option....176
EOLIN - SET/QUERY/EXTRACT Option....177
EOLOUT - SET/QUERY/EXTRACT Option....178
FIELD - EXTRACT Option....179
FILEID - SET/QUERY/EXTRACT Option....179
FMODE - SET/QUERY/EXTRACT Option....179
FNAME, MBR - SET/QUERY/EXTRACT Option....179
FOCUS - EXTRACT Option....180
FORMAT - SET/QUERY/EXTRACT Option....180
FPATH - SET/QUERY/EXTRACT Option....181
FTYPE - SET/QUERY/EXTRACT Option....181
FVALUE - EXTRACT Option....181
IDSCOPE - SET/QUERY/EXTRACT Option....182
IDWARNING - SET/QUERY/EXTRACT Option....183
KEY - QUERY/EXTRACT Option....183
LASTMSG - QUERY/EXTRACT Option....184
LENGTH - SET/QUERY/EXTRACT Option....184
LOADWARNING - SET/QUERY/EXTRACT Option....185
LRECL - SET/QUERY/EXTRACT Option....186
MACROPATH - SET/QUERY/EXTRACT Option....186
MAPPING - SET/QUERY/EXTRACT Option....186
MAXCOBOLRC - SET/QUERY/EXTRACT Option....187
MAXPL1RC - SET/QUERY/EXTRACT Option....187
MAXSTOR - SET/QUERY/EXTRACT Option....188
MSGLINE - SET/QUERY/EXTRACT Option....189
MSGMODE - SET/QUERY/EXTRACT Option....190
MULTIPOINT - SET/QUERY/EXTRACT Option....190
OFFSET - SET/QUERY/EXTRACT Option....191
PAD - SET/QUERY/EXTRACT Option....192
PFKEY - SET/QUERY/EXTRACT Option....193
POINT - SET/QUERY/EXTRACT Option....193
PREFIX - SET/QUERY/EXTRACT Option....194
QSEPARATOR - QUERY/EXTRACT Option....195
RECFM - SET/QUERY/EXTRACT Option....195
RECINFO - SET/QUERY/EXTRACT Option....195
RECTYPES - QUERY/EXTRACT Option....198
REFERENCE - SET/QUERY/EXTRACT Option....198
REGION - QUERY/EXTRACT Option....199
RESERVED - SET/QUERY/EXTRACT Option....199
SCALE - SET/QUERY/EXTRACT Option....199
SESSION - QUERY/EXTRACT Option....200
SHADOW - SET/QUERY/EXTRACT Option....201
SIZE - QUERY/EXTRACT Option....202
TITLE - SET/QUERY/EXTRACT Option....202
TYPE - SET/QUERY/EXTRACT Option....203
UNDOING - SET/QUERY/EXTRACT Option....204
UNNAMED - SET/QUERY/EXTRACT Option....205
USERNAME - QUERY/EXTRACT Option....205
USING - QUERY/EXTRACT Option....206
VALUE - EXTRACT Option....206

Contents

SET/QUERY/EXTRACT
VIEW - SET/QUERY/EXTRACT Option....207
WINNAME - SET/QUERY/EXTRACT Option....207
WINPOS - SET/QUERY/EXTRACT Option....207
WINSIZE - SET/QUERY/EXTRACT Option....207
WRAP - SET/QUERY/EXTRACT Option....208

Prefix Area (Line) Commands....209

Function Keys....211

Glossary....212

Documentation Notes

First Edition, May 2012

Information in this document details general features and functionality of the **SELCOPY Product Suite 3.10** component, **SELCOPY/i**.

This document replaces the previous edition of *SELCOPY/i Structured Data Editor (SDE)* applicable to **SELCOPY Product Suite 3.00**, which is now obsolete.

Copyright in the whole and every part of this document and of the SELCOPY Products Suite system and programs, is owned by Compute (Bridgend) Ltd (hereinafter referred to as CBL), whose registered office is located at 8 Merthyr Mawr Road, Bridgend, Wales, UK, CF31 3NH, and who reserve the right to alter, at their convenience, the whole or any part of this document and/or the SELCOPY Product Suite system and programs.

SELCOPY Product Suite for z/OS, z/VM (CMS) and z/VSE operating systems, which includes SELCOPY, SELCOPY/i and CBLVCAT, is available for download and install from **http://www.cbl.com/selcdl.html**.

The following publications for SELCOPY Product Suite and its component products are available in Adobe Acrobat PDF format at CBL web page **http://www.cbl.com/selcdoc.html**:

- SELCOPY Product Suite Customisation Guide
- SELCOPY User Manual
- CBLVCAT User Manual
- SELCOPY/i Reference and User Guide
- SELCOPY/i Text Editor (CBLe) Manual
- SELCOPY/i Structured Data Editor Manual

No reproduction of the whole or any part of the SELCOPY Product Suite system and programs, or of this document, is to be made without prior written authority from Compute (Bridgend) Ltd.

At the time of publication, this document is believed to be correct. Where the program product differs from that stated herein, Compute (Bridgend) Ltd reserve the right to revise either the program or its documentation at their discretion. CBL do not warrant that upward compatibility will be maintained for any use made of this program product to perform any operation in a manner not documented within the user manual.

The following generic terms are used throughout this document to indicate all available versions and releases of IBM mainframe operating systems:

MVS - z/OS, OS/390, MVS/ESA, MVS/XA, MVS/SP, OS.

VSE - z/VSE, VSE/ESA, VSE/SP, DOS.

CMS - z/VM, VM/ESA, VM/XA, VM/SP.

All - All MVS, VSE and CMS operating systems.

Summary of Changes

This section is a summary of significant new features provided in SELCOPY/i Release 3.10.

First Edition (May 2012)

Segmented Record Support

Introduce SDE structured data edit and browse of data sets containing segmented records.

For details, see:

- *"Segmented Records"*
- *"Structure Definition and Record-Type Assignment"*
- *"Specifying USE WHEN Segment Identification Criteria"*
- *"Segmented Records SDE Browse/Edit View"*
- *"Navigating Segmented Record Display"*
- *"Segmented Record Edit"*
- CREATE STRUCTURE
- IDENTIFY
- SEGTYPE
- NEXT
- PREVIOUS

Record Filter Support

Introduce filter expression and filter data sets to be applied on SDE structured data edit and browse.

For details, see:

- *"Display a Subset of Records or Record Segments"*
- BROWSE
- EDIT

Enhancements to SDE Expressions

SDE expressions use to identify and select records have been improved to support qualified field referneces and built-in functions.

For details, see:

- *"Expressions"*
- *"Expression Terms"*
- *"Expression Operators"*
- *"Function Calls"*

Create SDO Structure panels

Introduce panels that generate an SELCOPY/i SDE structure (SDO) from multiple copy books or a File-Aid XREF PDS/PDSE library member.

For details, see:

- *"Display Records or Record Segments using multiple COBOL or PL/1 Copy Books"*

Enhancements SDE Data Edit panel

The menu items and input fields have been enhanced to support a record filter expression and COBOL REPLACE operation.

For details, see:

- *"Opening an SDE Edit/Browse View"*
- *"Display Records using a single COBOL or PL/1 Copy Book"*

HEXDUMP, CHAR, VFMT, MAP and UNFMT Support

Introduce commands HEXDUMP, CHAR, VFMT, MAP and UNFMT to format display of record data.

For details, see:

◊ *"Display Unformatted Record Data"*
◊ *"Displaying a Record/Segment in Single Record View"*
◊ *"Displaying Hexadecimal Representation of Record Data"*
◊ HEXDUMP
◊ CHAR
◊ MAP
◊ UNFMT
◊ VFMT

Miscellaneous Enhancements

The following miscellaneous operations have been introduced or enhanced:

◊ **IDENTIFY** primary command. Forces record type assignment for flagged records.
◊ Option **IDSCOPE**. Defines the scope of records affected by IDENTIFY.
◊ **ONLY** primary command. Exclude records that do not match a search string
◊ **FIND**, **EXCLUDE**, **CHANGE** primary commands. Support a relational operator.
◊ **LOCATE** primary command. Support location of field by name.
◊ **LOCATE**, **WHERE**, **MORE**, **LESS** primary commands. Support line flag keywords.
◊ **DISPLAY RECTYPES** primary command. List window of record-type definitions.
◊ **LIST SDO** primary command. PDS/PDSE library list of SDO structures.
◊ **LAYOUT** primary command. Display copy book layout of fields.

Introduction to SDE

Data set records may have an associated file structure that maps field information (position, length and data type) for all data within each data set record. These structures often exist as a PL/1 or COBOL copybook.

The SELCOPY/i Structured Data Environment (SDE) allows users to display and process structured data sets using a pre-defined SDE structure so that record data is formatted and arranged in field columns. An SDE structure may be generated from a copybook or using SDE's Create Structure internal syntax, and can contain a number of mappings, one for each different type of data set record.

SDE runs as a sub-component of the CBLe Text Editor so that window views displaying structured file data may be opened as child windows of a CBLe (or SELCOPY Interactive) MDI Frame window.

SELCOPY/i SDE enables users to do the following:

- Display, copy, edit and update structured data.
- Concurrently display records of more than one record type within the same view.
- Work with data in multiple or single record views.
- Select and order the fields displayed for each record type.
- Define new record structures using SELCOPY/i SDE syntax.

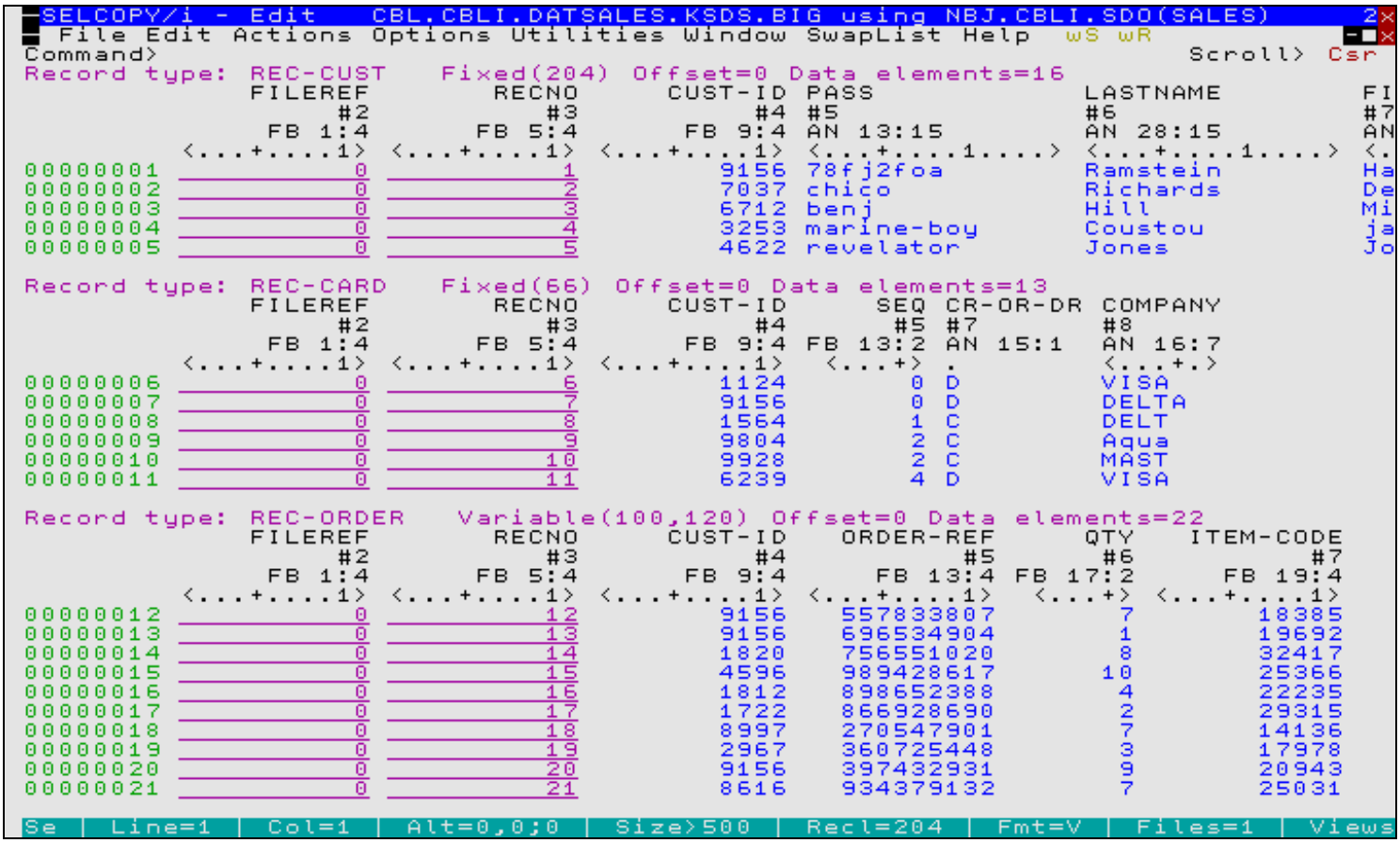


Figure 1. Structured Data Environment View.

Terminology

The following terminology is used throughout SELCOPY/i documentation in relation to record data processing in the SELCOPY/i SDE environment.

ADATA File
SYSADATA (Associated Data) file output generated by the Enterprise COBOL compiler when compile time option ADATA is specified, or by the Enterprise PL1 compiler when compile time option MSG, sub-option XINFO is specified.

SELCOPY/i SDE uses an ADATA file to generate an SDO.

COBOL Copybook
A data set or PDS/PDSE library member that contains one or more Enterprise COBOL group definitions (not necessarily 01-level) and is usually included by COBOL source programs to provide a file record layout.

Note that SELCOPY/i supports Enterprise COBOL only. Earlier releases of COBOL for z/OS do not support generation of ADATA output.

Expanded Record/Segment
A formatted record/segment that contains variable length fields or repeating groups.

In the process of formatting record data, Variable length fields are formatted as fields of length equal to the maximum length of the variable field. Repeating groups of fields are formatted so that a field group exists for the defined maximium number of occurrences of the repeating group.

All other types of record/segment are **unexpanded**.

Formatted Record/Segment
A typed record/segment for which the record data has been arranged into discrete fields as defined by the assigned RTO . An **unformatted** record/segment is one that is **untyped** or one that is **typed** with data arranged as a single character field of length equal to the record length.

Mapped Record/Segment
An alternative name for a formatted record/segment.
An **unmapped** record/segment is an alternative name for an untyped record/segment.

PL1 Copybook
A data set or PDS/PDSE library member that contains one or more Enterprise PL1 structure definitions and is usually included by PL1 source programs to provide a file record layout.

Record Structure
The definition of one or more fields that correspond to areas of a file record or DB2 table row or column. A record structure may define the complete or partial layout of fields within a file record or DB2 column.

Record-type
The name assigned to a single RTO definition in a SELCOPY/i SDO structure. Record-type is analogous with RTO and is often used to refer to an RTO definition.

RTO
A Record Type Object is a single record structure within an SDO. An RTO is referenced by a unique record-type name.

RTO Data Length
The sum of all field lengths defined by the RTO. The RTO data length may encompass a range of values if the RTO includes variable length fields or a variable number of repeating groups (e.g. array elements.)

SDO
A Structure Definition Object is a SELCOPY/i SDE generated structure, comprises one or more RTO and is used by SELCOPY/i SDE to map record data. If a COBOL/PL1 copybook or ADATA file is specified for record mapping, then SELCOPY/i will automatically generate an SDO from the copybook/ADATA source.

For this reason and unless otherwise stated, an SDO is often referenced simply as a structure. An SDO is referenced by its structure name which, if non-temporary, is the fileid of the structure definition disk file to which it is saved.

Segment or **Record Segment**
An area of record data treated as an individual record for Segmented browse/edit. Mapped segment boundaries are determined by the RTO data length and any offset defined for that RTO. An unmapped segment comprises all remaining data in the record that follows the last mapped segment.

Segmented Browse/Edit
A type of SDE file edit or browse that treats consecutive, non-overlapping areas (segments) of record data as individual records. Segmented browse/edit invoked when an SDO containing one or more RTO of type "SEC" is used to format record data.

Structure
A named object that contains one or more record structure. Unless otherwise stated, structure refers explicitly to a SELCOPY/i SDE structured data object (SDO).

Where stated, a structure may also apply to a COBOL or PL1 copybook or a COBOL or PL1 ADATA (Associated Data) compiler output file.

Typed Record/Segment

A record/segment that has an assigned record-type (RTO).
An **untyped** record/segment is one that has **not** been assigned an RTO .

Getting Started with SDE

The SELCOPY/i distribution includes sample structured data set members, COBOL copy books that map records within these members and the &PREFIX..CBLI.CMX(SDE) command centre (CMX) data set. The SDE CMX data set guides new users through some basic tasks related to structured data environment text editing, demonstrating each task by prompting the user to execute commands against the sample data.

It is strongly recommended that users take time to work their way through the SDE CMX data set in order to familiarise themselves with some of the features and capabilities of SDE edit.

Basic SDE edit tasks are also discussed in this section of the SDE documentation.

Handling Long Command Strings

In addition to using the supplied panel interface, all supported SDE tasks may also be performed using command line interface (CLI) commands.

By default, SELCOPY/i windows, including SDE window views, include a single command line for entering commands to be passed to the local Command Line Interface (CLI).

Whereas it is recommended that command strings, particularly command strings of any significant length, are executed from and stored for later use in the user's HOME (CMX command centre) file, a window's command line may be extended to allow input of long command strings.

The **Command Line** dialog window includes the "Lines>" field which defines the number of continuous command lines (between 1 and 9) to be presented in the current window. Having updated this value and selected OK, the change will take immediate effect.

The Command Line dialog window is opened by selecting the "Command Line ..." option of the system menu, (select the system menu button to the left of the window's title bar). Alternatively, execute the COMMANDLINE SELCOPY/i CLI command (synonym **CLN**).

Any text entered on second and subsequent command lines is appended to text on the command line before.

Opening an SDE Edit/Browse View

Basic tasks which relate to starting an SDE edit or browse view for processing structured data are discussed below.

- Display Records without a Structure
- Display Records using a single COBOL or PL/1 Copy Book
- Display Records or Record Segments using multiple COBOL or PL/1 Copy Books
- Display a Subset of Records or Record Segments

Display Records without a Structure

The simplest application of the structured data editor (SDE) is to display record data for edit or browse without applying a structure to format the record data into individual fields. Instead, records are displayed as a single character field of record-type and field name "UnMapped".

SDE edit of a file without application of a structure is an alternative to performing CBLe text edit on the file. Reasons for using SDE data edit are detailed below . Note that SDE browse is performed for the CBLe text editor BROWSE command.

To perform this type of edit or browse, do the following:

1. Open the SDE - Structured Data Browse/Edit panel by selecting **Data Edit** from the Primary Option Menu, or simply entering command SDE at any command prompt.

2. Enter the fileid of your data set, HFS file or PDS/PDSE library member in the appropriate INPUT field(s), select Browse or Edit options, then press <Enter> (or select menu item "Run").

   Note that the USING panel fields should be de-selected to bypass use of a structure.

Alternatively, use the SDE BROWSE or EDIT commands to open the SDE view directly. e.g.

```
<SDATA  EDIT  CBL.X2011036.JS03XAU.KSDS
<BROWSE    /u/smpe/smpnts/CBL12075/S0001.CBL.PROD.CBL11091.README
```

SDE Data Edit vs. CBLe Text Edit

Whereas, in most cases, use of the SELCOPY/i text editor (CBLe) to edit small, non-VSAM files is preferred, the SDE structured data editor offers some advantages over CBLe.

1. **Automatic selection of optimal EDIT technique.**
   SDE EDIT techniques are:

   - **KSDS** for VSAM KSDS data sets.
   - **In-storage** for files small enough to be loaded entirely within a region of available storage.
   - **In-place** for edit/browse of file records that need not be loaded into storage until required for processing.
   - **Auxiliary** for files that are too large to be loaded entirely in available storage.

   CBLe Text Edit supports only In-storage edit, so excluding the ability to edit large data sets.

2. **Select the type of EDIT to be performed on the data.**
   SDE supports Full EDIT, allowing change of record length as well as move, insert, copy and delete of records, or the more restrictive Update-in-place EDIT, allowing only update of existing records without changing record length.

   The benefit of Update-in-place EDIT is that, unless record subsetting is performed on load (see below), file records need only be loaded into storage when required for display or CHANGE processing.

   CBLe Text Edit supports Full EDIT only.

3. **VSAM Edit.**
   SDE supports full VSAM editing capabilities including edit of VSAM data sets defined with the NOREUSE attribute and location of records by KEY, RBA or record number as appropriate.

   CBLe text edit of VSAM data sets is restricted to data sets defined with the REUSE attribute and consequently re-writes all records when the data set is saved.

4. **Subset on records to be loaded for edit.**
   SDE supports use of a FILTER expression, to select only records that satisfy the expression, and/or FROM/FOR to select a range of records from the file.

   Specifying FILTER, FROM and/or FOR forces In-storage EDIT technique and Update-in-place EDIT.

5. **Display records in single-record view.**
   SDE supports alternating between multi-record and single record format display. Single record format displays the focus record so that its data wraps over several lines of the display if necessary. This is particularly useful when editing long records.

   SELCOPY/i's multiple views of the same data means that records may be displayed in both multi-record and single record views simultanously.

   CBLe Text Edit supports multiple record views only.

6. **Preserve Record Lengths**
   Unlike the CBLe text editor which automatically strips trailing blanks from records belonging to variable length record files, SDE preserves a record's length, and so, any trailing blanks. In SDE edit, a record's length may only be updated using a CHANGE command or by manually changing the record length value in the "Length" field (displayed in multi-record format using the RECLEN command.)

7. **Search and Exclude Records**
   In addition to more function rich CHANGE, EXCLUDE, FIND and ONLY commands, SDE supports built-in functions for use in expressions on WHERE, MORE and LESS commands, for including and excluding records from the current display, and also on the LOCATE command for scrolling to individual records.

8. **Support for RRDS data set empty slots.**
   SDE supports insert of records into empty slots of an RRDS data set defined with REUSE.

Display Records using a single COBOL or PL/1 Copy Book

File records may be possess the same or different structures as defined by COBOL group items or PL/1 structures referenced by or within a single copybook data set or PDS/PDSE library member.

If this is the case, a formatted display of the file's records may be opened in an SDE edit or browse view simply by referencing the appropriate copy book file as described below.

1. Open the SDE - Structured Data Browse/Edit panel by selecting **Data Edit** from the Primary Option Menu, or simply entering command SDE at any command prompt.

2. Enter the fileid of your data set, HFS file or PDS/PDSE library member in the appropriate INPUT field(s).

3. Select Browse or Edit options and, for Edit, select the type of Edit to be performed.

4. Activate the "USING ____ Copybook:" option field by selecting it with "/", then enter the copy book type (COBOL or PL1) in the field between USING and Copybook.

5. Enter the fileid of the copy book in the input field following "USING ____ Copybook:" This may be a sequential file DSN or PDS/PDSE library DSN and member name.

6. For COBOL copy books only, if the copy book includes fields which comprise variable pseudo-text strings, then select "REPLACE" from the menu bar to open the COBOL Compiler Options panel. Review and, if necessary, add COBOL replacing options that convert the pseudo-text strings in your copy book.

7. Press <Enter> (or select menu item "Run") to open the SDE Edit/Browse view.

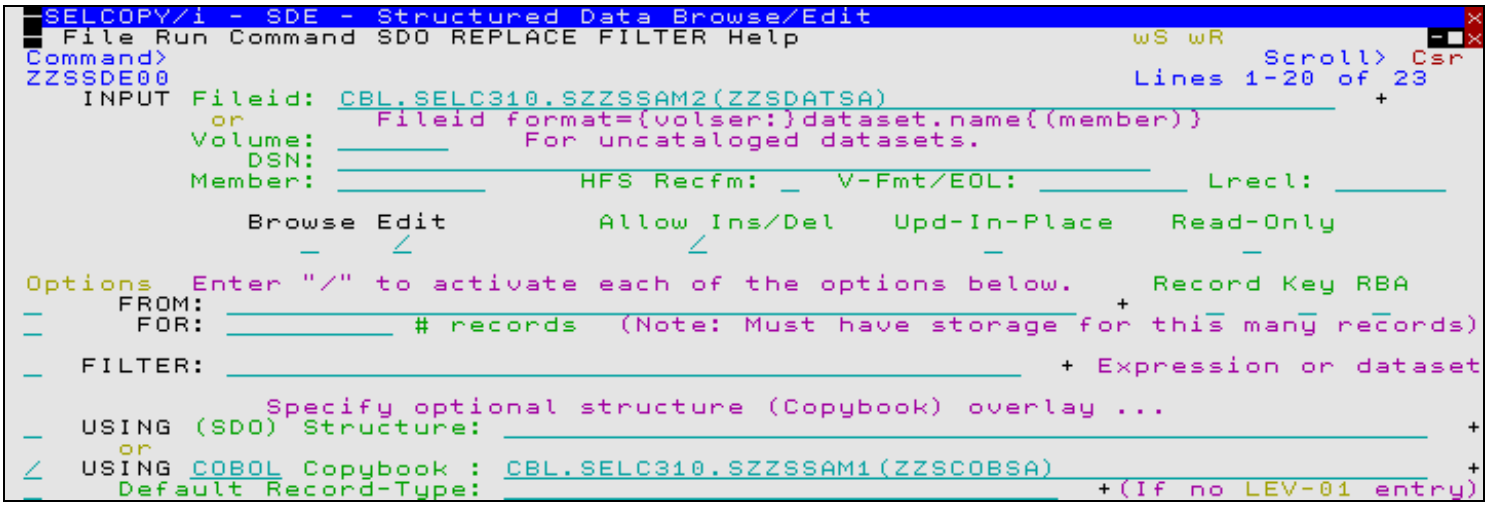

SELCOPY/i - SDE - Structured Data Browse/Edit
File Run Command SDO REPLACE FILTER Help
Command>
ZZSSDE00
Lines 1-20 of 23
INPUT Fileid: CBL.SELC310.SZZSSAM2(ZZSDATSA)
or Fileid format={volser:}dataset.name{(member)}
Volume: For uncataloged datasets.
DSN:
Member: HFS Recfm: V-Fmt/EOL: Lrecl:
Browse Edit Allow Ins/Del Upd-In-Place Read-Only
Options Enter "/" to activate each of the options below. Record Key RBA
FROM:
FOR: # records (Note: Must have storage for this many records)
FILTER: Expression or dataset
Specify optional structure (Copybook) overlay ...
USING (SDO) Structure:
or
USING COBOL Copybook : CBL.SELC310.SZZSSAM1(ZZSCOBSA)
Default Record-Type: (If no LEV-01 entry)

Figure 2. SDE - Structured Data Browse/Edit panel - USING Copybook.

In order to format records into discrete fields, SDE uses a structure definition object (SDO). An SDO is a pseudo-compiled object that may contain any number of record structure (record-type) definitions and may be generated automatically by SDE as part of an edit or browse operation, or directly using the SDE CREATE STRUCTURE command.

When using SDE to perform formatted record edit or browse using a COBOL or PL/1 copy book, SDE generates a temporary SDO which involves compiling the copy book.

Note: Since compilation is a processing overhead, it is recommended that, if the copy book is to be used regularly to format file record data (for Edit, browse, File search or File Compare), then a non-temporary SDO should created, saved and used by these operations instead of the copy book.

Create an SDE structure (SDO) from a single COBOL or PL/1 Copy Book

A non-temporary SDE structure definition object (SDO) may be generated and saved to a sequential data set or PDS/PDSE library member for subsequent use by any operation that formats structured record data. e.g. SDE BROWSE, EDIT, File Search/Update/Copy/Remap (FSU) and File Compare (COMPFILE).

If the SDO is to be generated from a single COBOL or PL/1 copy book source, then the SDE - Structured Data Browse/Edit panel may be used. If multiple source copy books are to be used for generating an SDO containing either multiple record structure (record-type) definitions or segmented record record-type definitions, then the Create Structure from COBOL/PL1 Copybook(s) panel should be used. (See *"Create an SDE structure (SDO) from a multipple COBOL or PL/1 Copy Books"*.)

To generate an SDO from a single copy book source:

1. Open the **SDE - Structured Data Browse/Edit panel** as described previously.

2. Enter the fileid of the SDO file to be created in the input field following "USING (SDO) Structure:" This may be a sequential file DSN or PDS/PDSE library DSN and member name.
   Note that the field option does not need to be activated.

3. In field option "USING ____ Copybook:", enter the copy book type (COBOL or PL1) in the field between USING and Copybook.
   Note that the field option does not need to be activated.

4. Enter the fileid of the copy book in the input field following "USING ____ Copybook:"

5. For COBOL copy books only, if the copy book includes fields which comprise variable pseudo-text strings, then select "REPLACE" from the menu bar to open the COBOL Compiler Options panel. Review and, if necessary, add COBOL replacing options that convert the pseudo-text strings in your copy book.

6. Select menu item "SDO" to generate the SDO and save it to disk using the specified SDO fileid.

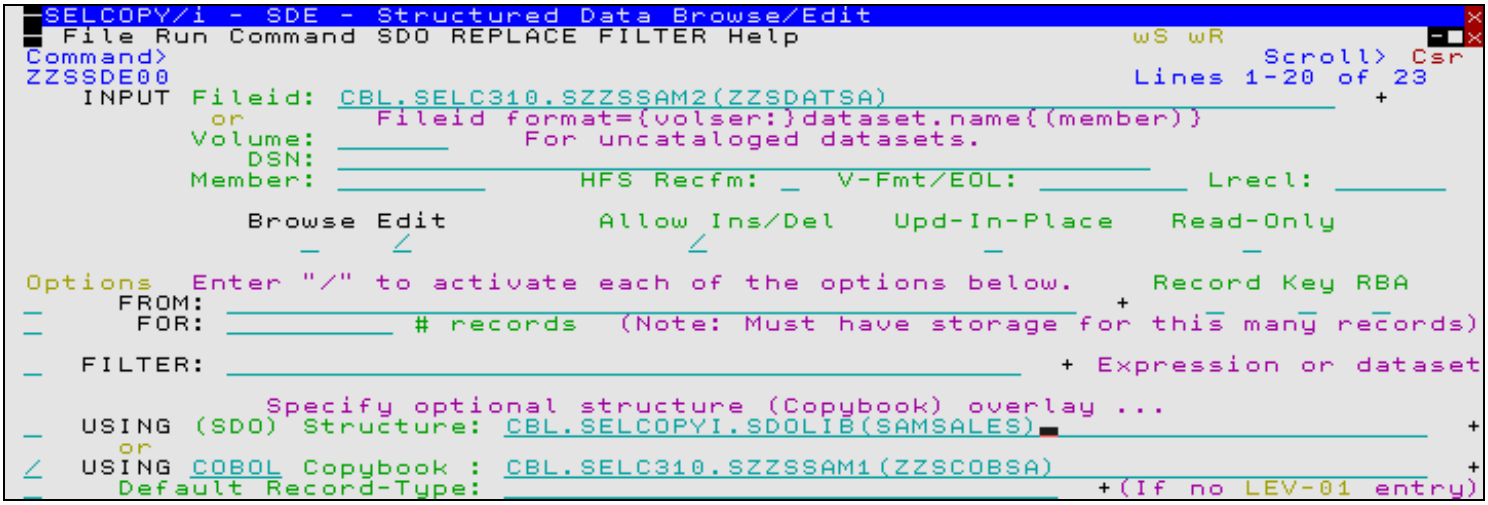

SELCOPY/i - SDE - Structured Data Browse/Edit
File Run Command SDO REPLACE FILTER Help
Command>
ZZSSDE00
Scroll> Csr
Lines 1-20 of 23
INPUT Fileid: CBL.SELC310.SZZSSAM2(ZZSDATSA)
or Fileid format={volser:}dataset.name{(member)}
Volume:
DSN:
For uncataloged datasets.
Member:
HFS Recfm: _ V-Fmt/EOL: Lrecl:
Browse Edit Allow Ins/Del Upd-In-Place Read-Only
Options Enter "/" to activate each of the options below. Record Key RBA
FROM:
FOR: # records (Note: Must have storage for this many records)
FILTER: + Expression or dataset
Specify optional structure (Copybook) overlay ...
USING (SDO) Structure: CBL.SELCOPYI.SDOLIB(SAMSALES)
or
USING COBOL Copybook : CBL.SELC310.SZZSSAM1(ZZSCOBSA)
Default Record-Type: +(If no LEV-01 entry)

Figure 3. SDE - Structured Data Browse/Edit panel (Create SDO).

Alternatively, use the SDE CREATE STRUCTURE command to generate the SDO via the command line interface. e.g.

```
<SDATA  CREATE STRUCT       CBL.SELCOPYI.SDOLIB(SAMSALES)     \
              FROM  COBOL   CBL.SELC310.SZZSSAM1(ZZSCOBSA)
```

Note:

1. The CREATE STRUCTURE command may be executed from within SELCOPY/i via the command line or from a CBLe text edited file using CMDTEXT (<PF4>). Additionally, it may be included in SDEIN input for batch execution using the SDEAMAIN executable program.
2. Unlike the SDE Edit/Browse panel method, generating an SDO using CREATE STRUCTURE command supports specification of multiple copy book files, each potentially containing one or more record structure definitions (group items).

Create an SDE structure (SDO) Compiler Return Code

SELCOPY/i SDE compiler options exist that nominate the compiler load module and also the maximum compiler return code for which SDE will continue to generate an SDO.

If the SDO generation fails with the following message then an unexpected return code has been set by the compiler and so SELCOPY/i displays the compiler output listing in a text edit view.

```
ZZSD087E COBOL/PL1 compile has failed with return code nn (decimal). To update
 your maximum acceptable return code (currently value=0) select "Settings"
 from the Primary Option Menu (=). Then select "Data Edit" options.
```

Review the compiler output listing to discover the cause of the compile error and, if possible, correct the error before re-running the SDO generation.

If, however, the cause of the return code set by the compiler is acceptable and SELCOPY/i should continue in its attempt to generate an SDO, then open the relevant compiler options panel as described in the message and ammend the "Max RC" value as appropriate.

Re-running the SDO generation may succeed or fail depending on the cause of the compiler return code.

Display Records or Record Segments using multiple COBOL or PL/1 Copy Books

In many cases, individual file records or segments within records have structures, defined by different COBOL group items or PL/1 structures, where each structure definition is saved in a separate copybook library member.

To format data in these types of files, an SDO must first be generated to consolidate the relevant copy book structure definitions and also define the record-type (structure definition) selection criteria. Data within a file's record, referenced by an individual record-type's selection criteria expression, determines whether that record-type is a match for the record or record segment data. If so, that record-type gets assigned to that record or record segment.

Create an SDE structure (SDO) from multiple COBOL or PL/1 Copy Books

To generate a non-temporary SDO from multiple COBOL and/or PL/1 copy book source PDS/PDSE members:

1. Open the Create Structure from COBOL/PL1 Copybook(s) panel by selecting **Structure** then **Copybook** from the Primary Option Menu, or simply entering command **SDO** at any command prompt.

2. Enter the fileid and, optionally, a title and description for the SDO PDS/PDSE library member to be created. To include a short title or description in the SDO definition, activate the "Title" and/or "Description" fields by entering "/" in the selection fields as required.

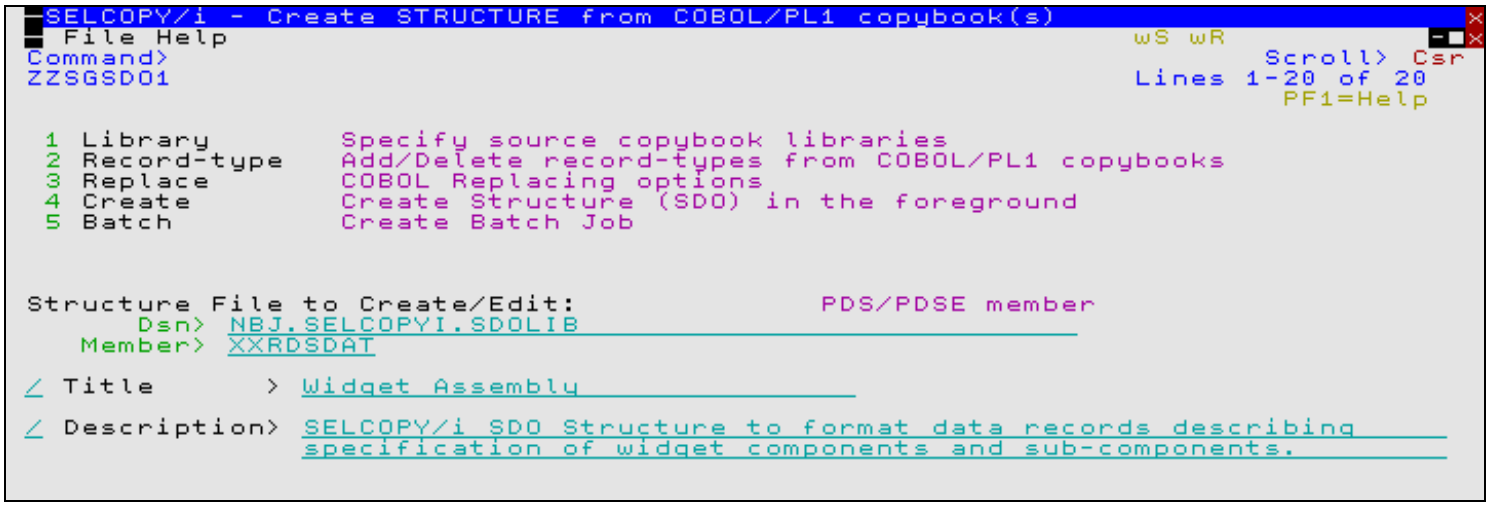

SELCOPY/i - Create STRUCTURE from COBOL/PL1 copybook(s)
File Help
wS wR
Command>
Scroll> Csr
ZZSGSD01
Lines 1-20 of 20
PF1=Help
1 Library Specify source copybook libraries
2 Record-type Add/Delete record-types from COBOL/PL1 copybooks
3 Replace COBOL Replacing options
4 Create Create Structure (SDO) in the foreground
5 Batch Create Batch Job
Structure File to Create/Edit: PDS/PDSE member
Dsn> NBJ.SELCOPYI.SDOLIB
Member> XXRDSDAT
/ Title > Widget Assembly
/ Description> SELCOPY/i SDO Structure to format data records describing specification of widget components and sub-components.

Figure 4. SDO - Create Structure from COBOL/PL1 Copybook(s).

3. Select option 1. "Library" to open the "Copybook Library List" panel view.

   This panel view contains an embedded table which supports a number of primary and line (prefix) commands including I (insert), R (replicate) and D (Delete).

   Insert an entry in the Copybook Library Dataset name for each copybook PDS/PDSE library containing the source copybooks for this SDO. Note that the order in which the library DSNs occur, defines the library search order for copybook member names.

   When all required library DSNs have been entered, press <PF3> to return to the main panel view.

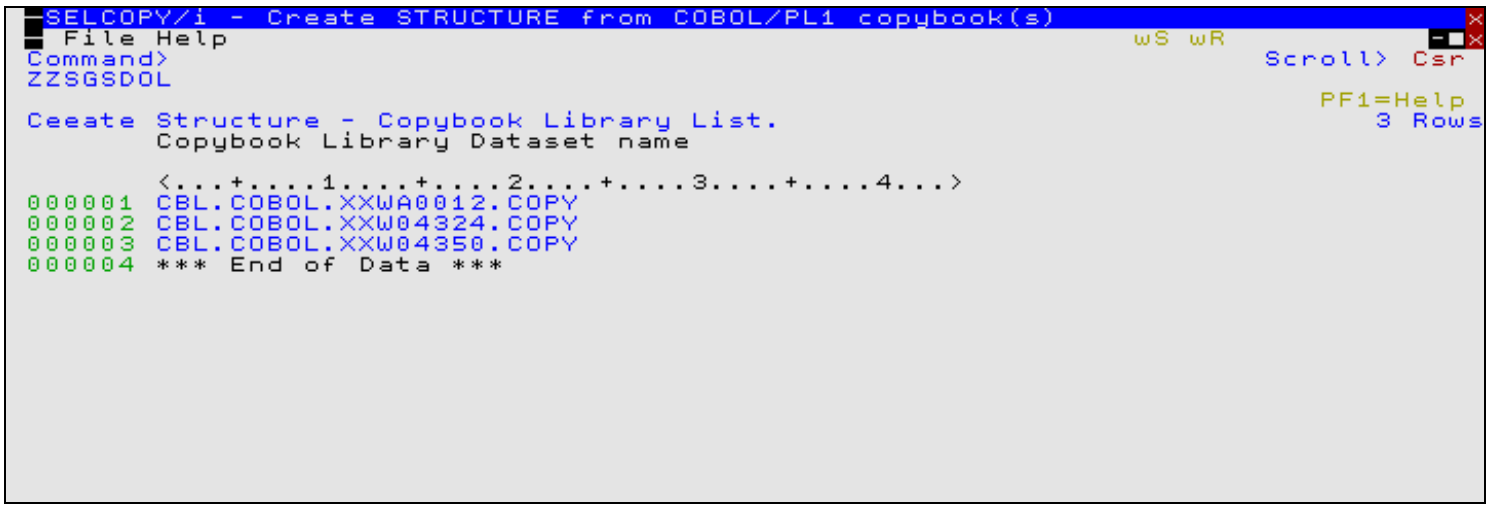


Figure 5. SDO - Create Structure from COBOL/PL1 Copybook(s) - (Library List).

4. Select option 2. "Record-Type" to open the "Define Record-Types" panel view.

   This panel view also contains an embedded table which should be updated first to contain a row for each copy book member record structure (record-type) definition included in the SDO.

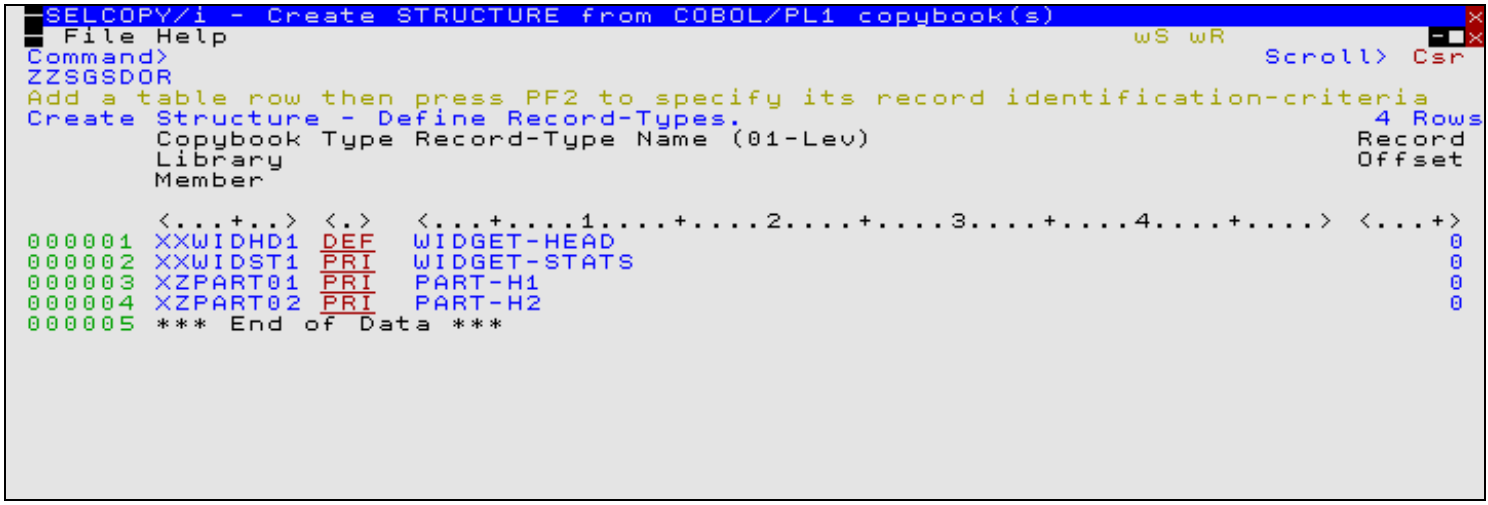


Figure 6. SDO - Create Structure from COBOL/PL1 Copybook(s) - (Record-Type List).

5. For each copybook member name in the table, place the cursor anywhere in the table row and press <PF2> to display all associated field entries for that copy book.

   Update associated fields as follow before pressing <PF2> to return to the "Define Record-Types" view and repeating the process for the next copybook entry.

   1. In the "Name" field, enter the name assigned to the required COBOL group item or PL/1 structure to be used to format record data and which exists in the copy book. (In a COBOL copybook, this is usually a 01 level group item.)

   2. Specify the "Type" (PRI, SEC or DEF) assigned to the record-type generated for this record structure. Any invalid value in this field will display a list of selectable entries.

      ◊ PRI indicates that the structure applies to an entire record or a primary (base) segment in a segmented record.
      ◊ SEC applies only to segmented records and indicates that the structure applies to a secondary segment in a segmented record.
      ◊ DEF may be specified for one record-type definition only and indicates that the structure is to be applied to all records or primary record segments that do not satisfy the selection criteria of any other PRI record-type.

   3. Specify the source copy book "Language" (COBOL or PL1). Again, any invalid value in this field will display a list of selectable entries.

   4. Optioanally, specify an "Offset" into record or record segment data at which formatting will begin. This may be a positive or negative integer value with default zero (0).

   5. For record-types PRI or SEC, activate the "Id" with "/" and specify an SDE expression that must be satisfied in order for this record-type to be assigned to the record data.

      Selection criteria expressions may be very complex and you should first familiarise yourself with available built-in functions , e.g. substr(), basepos() and segpos(), formatted field value referencing techniques and operators supported by SDE expressions.

      Press <PF2> with the cursor in the "Id" field to expand it into a text edit view. Having completed edit of the selection criteria expression, press <PF3> to close the edit view and copy the text to the "Id" field. You may also use <PF10> and <PF11> to scroll the text in expandable input fields.

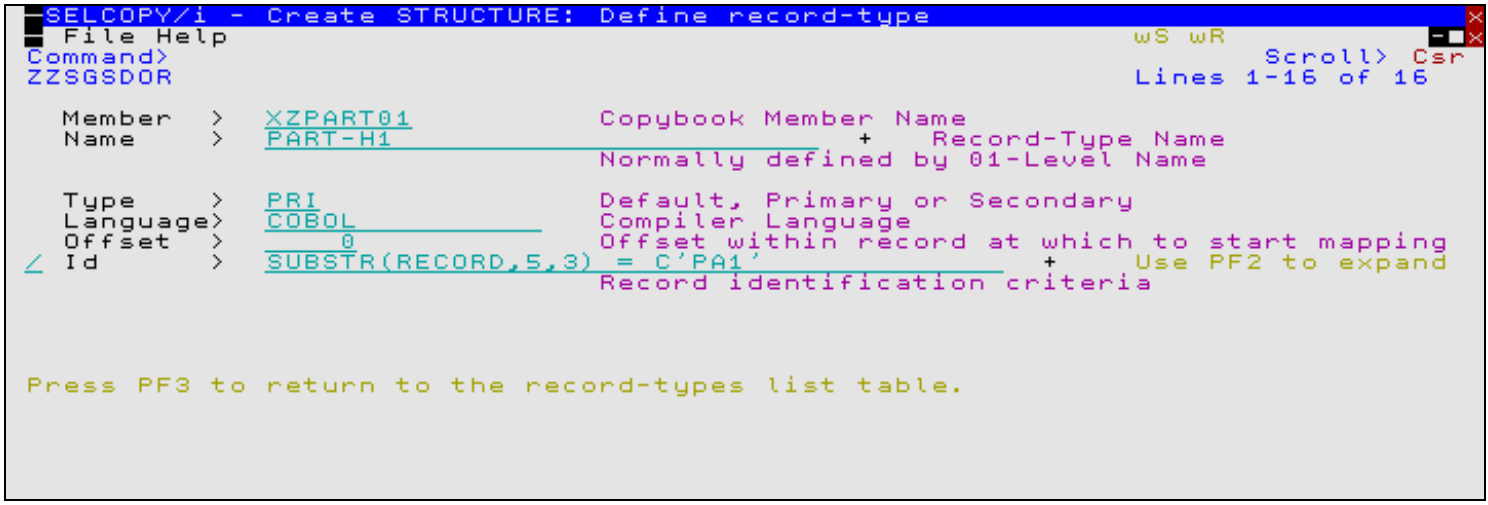

SELCOPY/i - Create STRUCTURE: Define record-type
File Help
wS wR
Command>
Scroll> Csr
ZZSGSDOR
Lines 1-16 of 16
Member > XZPART01 Copybook Member Name
Name > PART-H1 + Record-Type Name
Normally defined by 01-Level Name
Type > PRI Default, Primary or Secondary
Language> COBOL Compiler Language
Offset > 0 Offset within record at which to start mapping
Id > SUBSTR(RECORD,5,3) = C'PA1' + Use PF2 to expand
Record identification criteria
Press PF3 to return to the record-types list table.

Figure 7. SDO - Create Structure from COBOL/PL1 Copybook(s) - (Record-Type View).

6. When all record-type definitions have been configured, press <PF3> to return to the main panel view.
7. For COBOL copy books only, if any of the copy books include fields which comprise variable pseudo-text strings, then select option 3. "Replace" to open the COBOL Compiler Options panel. Review and, if necessary, add COBOL replacing options that convert the pseudo-text strings in your copy books.
8. Select option 4. "Create" to generate the SDO in the foreground. Alternatively, select option 5. "Batch" to create an SDEAMAIN batch job for generating the SDO and display it in a text edit view.

Display Records or Record Segments using an SDO structure

The fileid of the saved SDO may be specified in the "USING (SDO) Structure" input field of the SDE - Structured Data Browse/Edit panel for any Edit or Browse of the file containing the record data to which it applies. Note that the same SDO may be used to format the file's records in other SELCOPY/i applications including Formatted File Search, Update, Remap and also Formatted or Hierarchical File Compare .

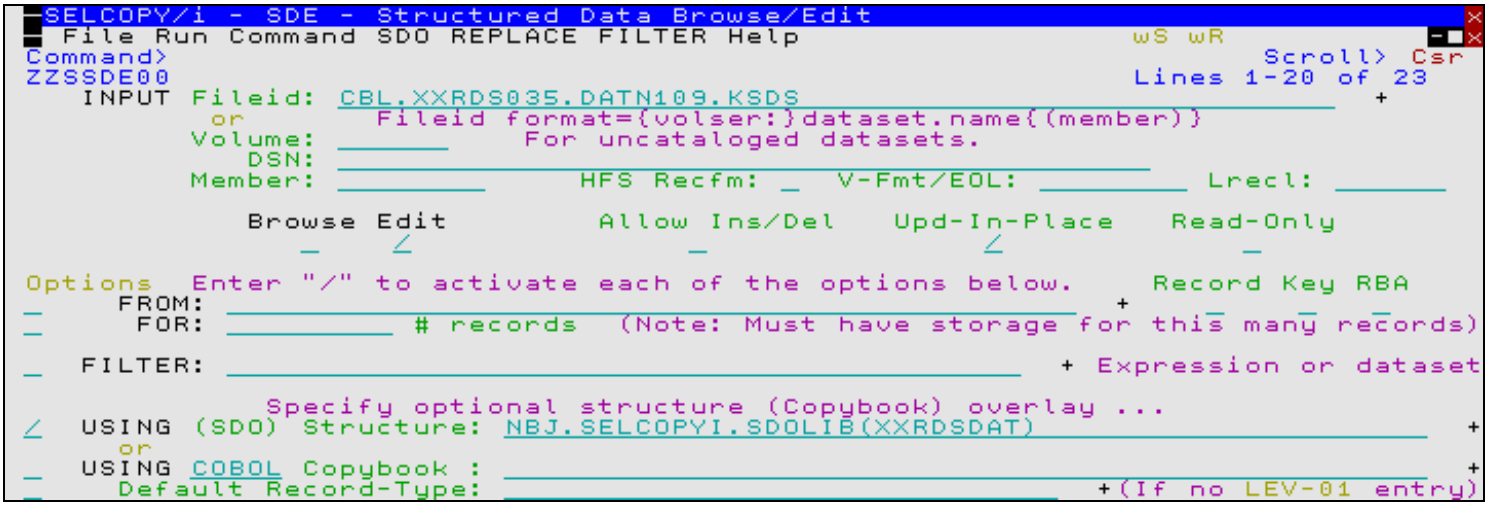

SELCOPY/i - SDE - Structured Data Browse/Edit
File Run Command SDO REPLACE FILTER Help
wS wR
Command>
Scroll> Csr
ZZSSDE00
Lines 1-20 of 23
INPUT Fileid: CBL.XXRDS035.DATN109.KSDS +
or Fileid format={volser:}dataset.name{(member)}
Volume: For uncataloged datasets.
DSN:
Member: HFS Recfm: V-Fmt/EOL: Lrecl:
Browse Edit Allow Ins/Del Upd-In-Place Read-Only
Options Enter "/" to activate each of the options below. Record Key RBA
FROM: +
FOR: # records (Note: Must have storage for this many records)
FILTER: + Expression or dataset
Specify optional structure (Copybook) overlay ...
USING (SDO) Structure: NBJ.SELCOPYI.SDOLIB(XXRDSDAT) +
or
USING COBOL Copybook : +
Default Record-Type: +(If no LEV-01 entry)

Figure 8. SDE - Structured Data Browse/Edit panel - USING SDO.

Display a Subset of Records or Record Segments

A specific subset of records may be selected for Browse or Edit from within a file using the FROM, FOR and/or FILTER field entries in the **SDE - Structured Data Browse/Edit** panel.

Note, however, that subsetting on records for either Edit or Browse will force In-storage record management (i.e. all selected records are loaded into storage.) Furthermore, for Edit only, the Update-In-Place Edit technique will be used, regardless of the edit technique selected by the user.

Subset using FROM and/or FOR

The **FROM** and **FOR** values identifies the first record and maximum number of records to be selected for display respectively.

If no FROM value is specified, the default first record will be the first record in the file. If no FOR value is specified, then there is no imposed limit to the number of records displayed.

Depending on the format of the file, the FROM value may identify the first record by means of a record number (decimal or hex numeric value), VSAM KSDS record key (character or hex string) or VSAM ESDS relative byte address (decimal or hex numeric value). The method of record identification to be used must be indicated by selecting one of the mutually exclusive options: Record, Key or RBA respectively.

FROM and FOR may be used in conjunction with FILTER so that the record filter only applies to records first selected by the FROM and/or FOR combination.

Create a record FILTER dataset

The **FILTER** input field accepts a temporary SDE expression or the fileid of a saved file containing an SDE expression. The expression when applied to a set of records (or record segments) tests either true or false, and so either selecting or deselecting the record as appropriate.

Particularly if a filter is to be used for subsequent edit or browse of a file, it is recommended that the filter expression is saved to a filter file.

To generate a filter file, either edit a new sequential data set, HFS file or PDS/PDSE library member and insert Filter Clause syntax, or use the Create File Filter panel.

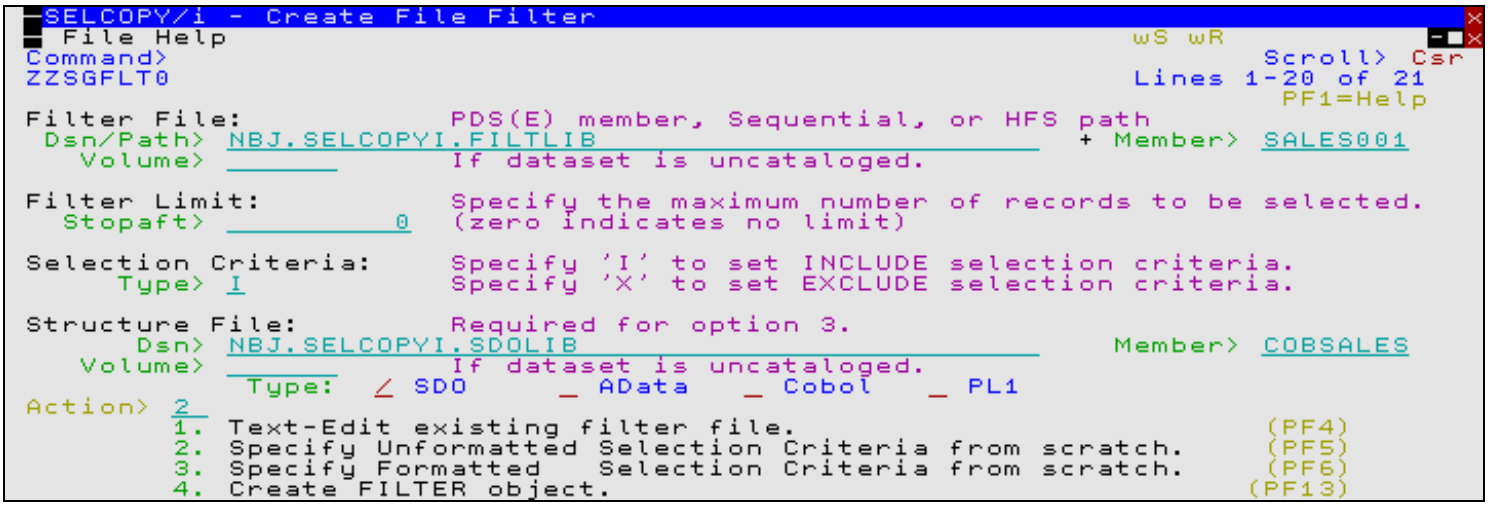


Figure 9. Create File Filter.

1. Open the **Create File Filter** panel by selecting **Filter** from the Primary Option Menu, the **FILTER** menu bar item from the **SDE - Structured Data Browse/Edit** panel or simply by entering command **FILTER** at any command prompt.
2. Insert the filter fileid in the **Filter File** input fields.
3. In the **Selection Criteria Type** field, enter either "I" or "X" to indicate whether records that satisfy the filter expression are to be included or excluded respectively.
4. If the filter is to be applied to only a limited number of records, after which no further filtering occurs, then enter a maximum number of records value in the **Filter Limit** field.
5. The filter may either reference data within the unformatted record or data within fields located at a fixed position of the formatted record (i.e. the field does not follow a variable length field or repeating group.) However, a filter may not combine the two methods.

   If the filter is to reference unformatted record data, then select option 2. "Specify Unformatted Selection Criteria from scratch" to build a single filter expression.

   If the filter is to reference formatted record data fields, then:

   1. If the **Structure File** input fileds are not already configured, enter the fileid and type of the structure file (SDO, COBOL/PL1 copybook, etc.) to which this filter will apply.
   2. Select option 3. "Specify Formatted Selection Criteria from scratch" to begin building filter expressions based on record-type definitions in the structure file.

6. Having configured filter expressions, select option 4. "Create FILTER object" to open a text edit view of the generated filter file containing filter syntax. Press <PF3> to save this file and again to exit the Cretae Filter File panel.

Filter Unformatted Record Data

The **FILTER (unformatted) - Selection Criteria** panel view contains an embedded table which supports a number of primary and line (prefix) commands including I (insert), R (replicate) and D (Delete).

Using these table editing commands, insert a row entry in the table for each logical sub-expression in the order in which they are to appear within the filter expression.

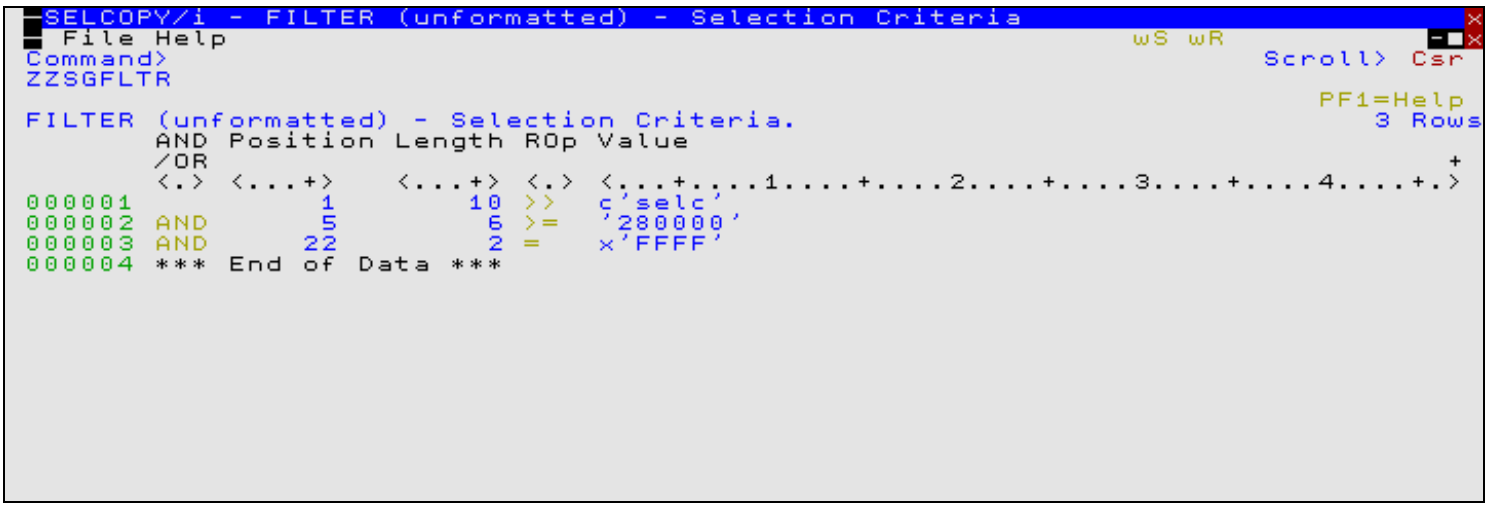


Figure 10. Create File Filter - Unformatted.

Each table row should be updated as follows:

1. Each sub-expression has a logical association with the preceeding sub-expression, represented by logical operator "AND" or "OR".

   Except for the first row entry in the table which has no no preceeding sub-expression, insert the appropriate logical operator in the **AND**/**OR** column. The required logical operator may be selected from a list of supported operators by entering a blank or invalid entry in this column.

   Note that support of "( )" parentheses, to override the standard precedence for evaluating expressions containing both AND and OR operators, is not yet supported by the Create Filter panel. If parentheses are required, they must be inserted manually in the generated filter expression.

2. Enter the position and length of the unformatted record data to be tested, in the **Position** and **Length** columns respectively.

3. Enter the relational operator used by the sub-expression in the **ROp** column. The required relational operator may be selected from a list of supported operators by entering a blank or invalid entry in this column.

4. Enter a character literal string value in the **Value** column against which the record data will be compared.

If required, a table row entry may be displayed in a formatted single row view by pressing <PF2> on the row required. The table column entries become input panel fields when in single row view. <PF3> returns to the table view.

When all required sub-expressions have been entered, press <PF3> to return to the main Create Filter File panel view.

Filter Formatted Record Data

The **Create Filter - Generate Include/Exclude Subclause(s)** panel view contains an embedded table displaying all record-types defined in the selected structure file.

For reference purposes only, the table includes column **Identification Criteria** to display any defined record-type identification criteria.

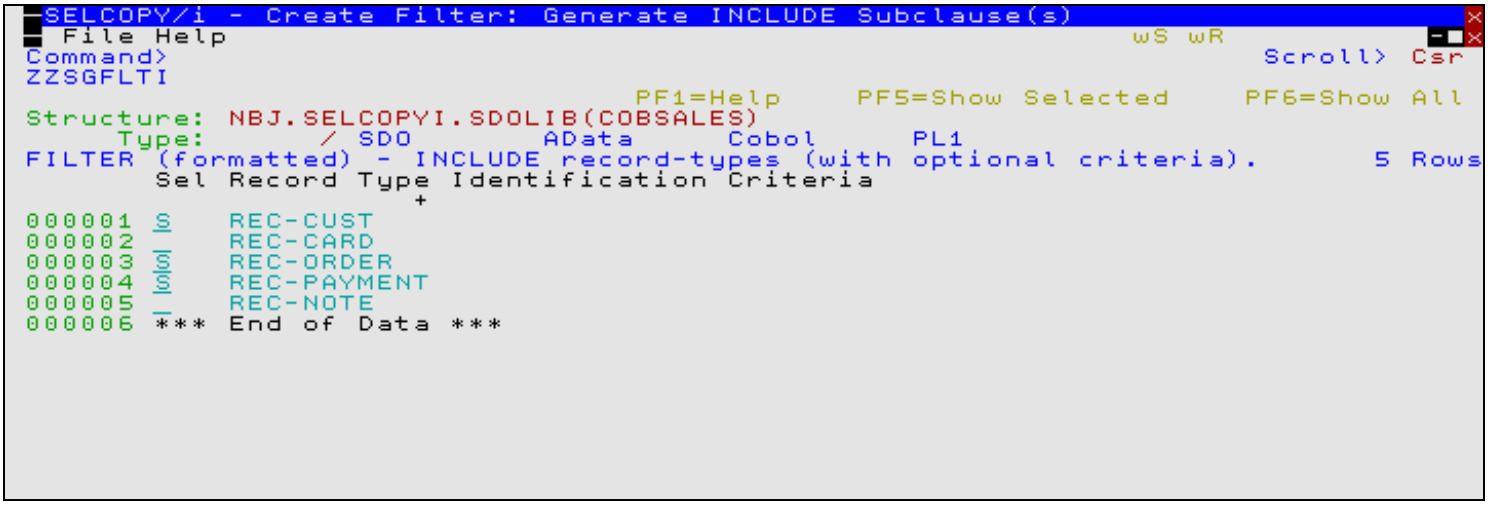


Figure 11. Create File Filter - Formatted Record-Types.

Select the record-types for which a filter sub-clause expression will be defined by entering "S" (or any non-blank character) in the **Sel** column for the required entries. Note that multiple filter sub-clauses may be generated the same record-type by duplicating the row referencing that record-type and selecting it.

Records assigned record-types that are not selected by the filter, are not interrogated when the filter is applied. If the filter includes records, then these records will be excluded. If the filter excludes records, then these records will be included.

On pressing <Enter>, the **Filter (formatted) - Selection Criteria** panel view will be opened for each selected record-type entry.

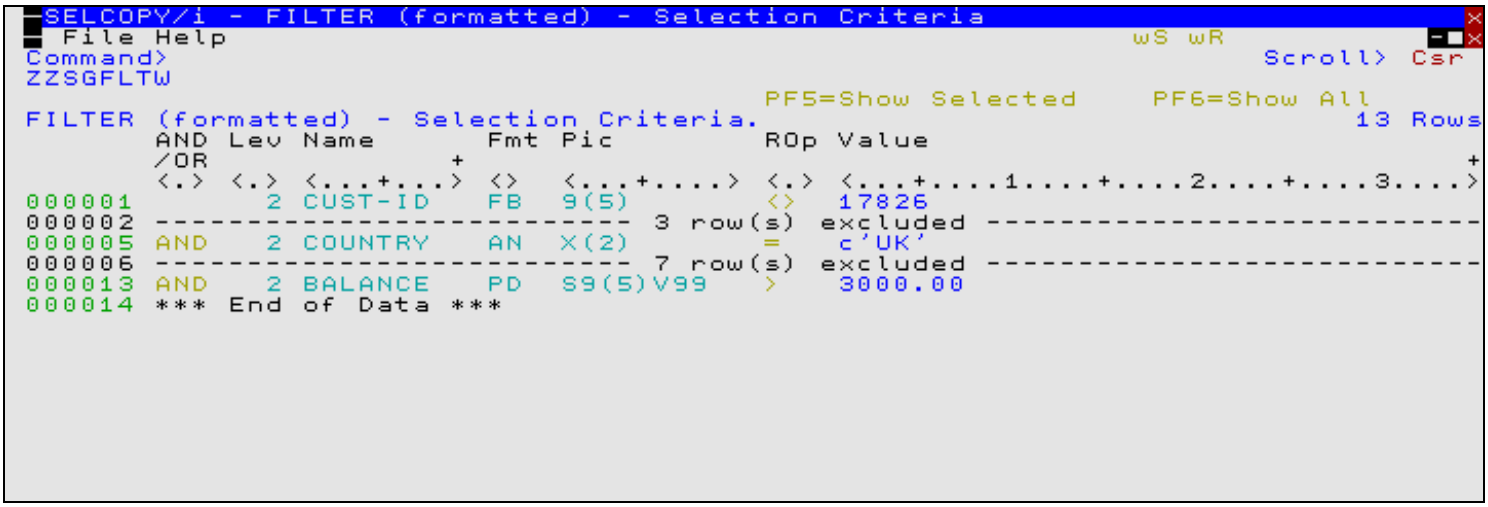

SELCOPY/i - FILTER (formatted) - Selection Criteria
File Help
wS wR
Command>
Scroll> Csr
ZZSGFLTW
PF5=Show Selected PF6=Show All
FILTER (formatted) - Selection Criteria. 13 Rows
AND Lev Name Fmt Pic ROp Value
/OR
000001 2 CUST-ID FB 9(5) <> 17826
000002 ---------- 3 row(s) excluded ----------
000005 AND 2 COUNTRY AN X(2) = c'UK'
000006 ---------- 7 row(s) excluded ----------
000013 AND 2 BALANCE PD S9(5)V99 > 3000.00
000014 *** End of Data ***

Figure 12. Create File Filter - Formatted Fields.

This panel view also contains an **embedded table** displaying all fields defined in the selected record-type. Like an unformatted record filter, a filter sub-clause (expression) for records assigned one of the selected record-types, comprises one or more sub-expressions. Each sub-expression being associated with the preceeding sub-expression via the logical operator, "AND" or "OR".

The table of fields should be updated as follows:

1. Use the "X" (or "XX") line command to exclude those fields **not** included as part of the filter expression for this record-type.

   If no fields are included, then the generated filter will simply include or exclude records assigned this record-type, as indicated by the type of filter (Include or Exclude).

2. Except for the first row entry in the table which has no no preceeding sub-expression, insert the appropriate logical operator in the **AND**/**OR** column. The required logical operator may be selected from a list of supported operators by entering a blank or invalid entry in this column.

   Note that support of "( )" parentheses, to override the standard precedence for evaluating expressions containing both AND and OR operators, is not yet supported by the Create Filter panels. If parentheses are required, they must be inserted manually in the generated filter expression. Otherwise, duplicate the record-type entry in the table of record-types and enter an alternate (OR) filter expression for that record-type.

3. In the **ROp** column, enter a relational operator suited to the datatype of the field. The required relational operator may be selected from a list of supported operators by entering a blank or invalid entry in this column.

4. In the **Value** column, enter a character literal string, Numerical value or Field value as appropriate to the datatype of the field. e.g. for a Binary numeric field (FB), Value must be numeric or reference a numeric field.

If required, a table row entry may be displayed in a formatted single row view by pressing <PF2> on the row required. The table column entries become input panel fields when in single row view. <PF3> returns to the table view.

When all required sub-expressions have been entered, press <PF3> to proceed to the **Filter (formatted) - Selection Criteria** panel for the next selected record-type, or return to the list of record-types view.

When expressions have been configured for each of the selected record-types, press <PF3> to return to the main Create Filter File panel view.

Subset using FILTER

The **FILTER** input field of the **SDE - Structured Data Browse/Edit panel** may contain filter syntax or the fileid of a filter file used to to selectively include or exclude records for display.

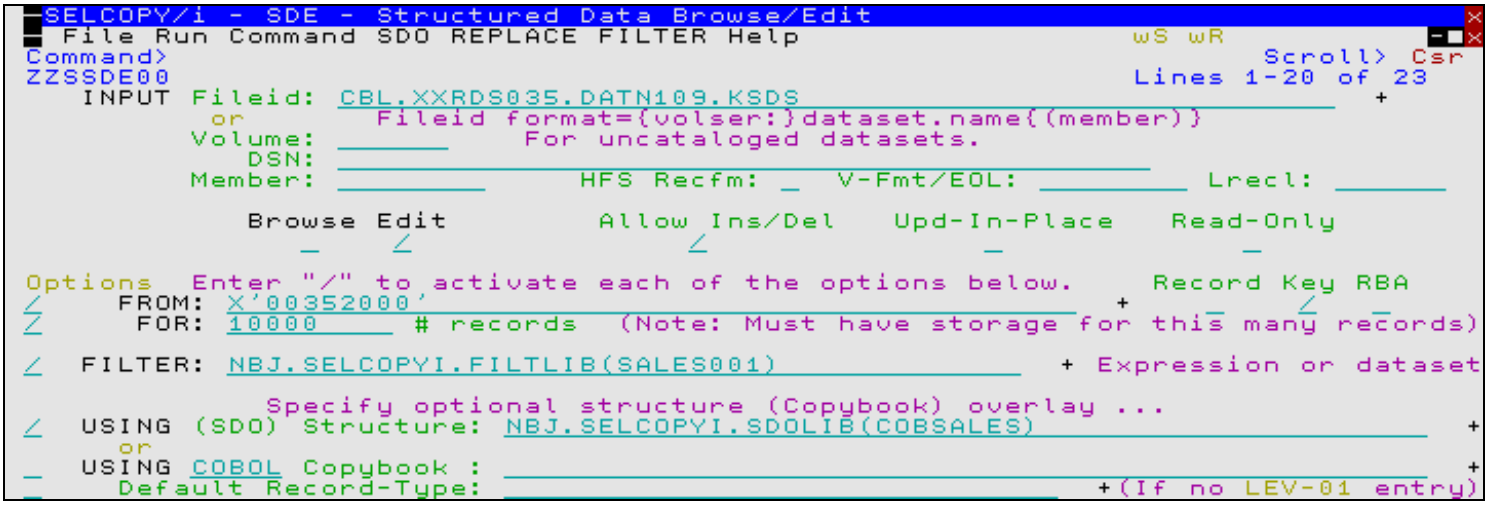

SELCOPY/i - SDE - Structured Data Browse/Edit
File Run Command SDO REPLACE FILTER Help
Command>
ZZSSDE00
Scroll> Csr
Lines 1-20 of 23
INPUT Fileid: CBL.XXRDS035.DATN109.KSDS
or Fileid format={volser:}dataset.name{(member)}
Volume: For uncataloged datasets.
DSN:
Member: HFS Recfm: V-Fmt/EOL: Lrecl:
Browse Edit Allow Ins/Del Upd-In-Place Read-Only
Options Enter "/" to activate each of the options below. Record Key RBA
FROM: X'00352000'
FOR: 10000 # records (Note: Must have storage for this many records)
FILTER: NBJ.SELCOPYI.FILTLIB(SALES001) Expression or dataset
Specify optional structure (Copybook) overlay ...
USING (SDO) Structure: NBJ.SELCOPYI.SDOLIB(COBSALES)
or
USING COBOL Copybook :
Default Record-Type: (If no LEV-01 entry)

Figure 13. SDE - Structured Data Browse/Edit panel - FILTER.

Changing the Record Display

The user can tailor the appearance of the SDE window view and the records displayed within.

An SDE window view is an MDI child window and can be resized and positioned using standard window manipulation as documented in the " *SELCOPY/i Reference and User Guide*". General manipulation of the SDE window contents is discussed in the following:

- Displaying Records of Specific Record Type(s)
- Display Unformatted Record Data
- Displaying a Record in Single Record View
- Displaying Hexadecimal Representation of Record Data
- Excluding Records from the Display
- Displaying Selected Columns
- Displaying the Record Headers

The record data may be displayed as Typed and/or Formatted, in Single or Multi record view, and/or with Hexadecimal or Hexadecimal dump display set on.

The many methods by which data may be displayed in an SDE view is summarised by the following table:

Typed	Formatted	Single Record	Hex	HexDump	Command Combination
Yes	Yes	No	No	No	`VFMT; HEX OFF`
Yes	Yes	No	Yes	No	`VFMT; HEX ON`
Yes	Yes	Yes	No	No	`MAP; HEX OFF`
Yes	Yes	Yes	Yes	No	`MAP; HEX ON`
Yes	No	No	No	No	`VFMT; FORMAT CHAR`
Yes	No	No	Yes	No	`VFMT; FORMAT HEX`
Yes	No	Yes	No	No	`MAP; FORMAT CHAR`
Yes	No	Yes	Yes	No	`MAP; FORMAT HEX`
No	No	No	No	No	`CHAR; HEX OFF`
No	No	No	Yes	No	`CHAR; HEX ON`
No	No	Yes	No	No	`UNFMT; HEX OFF`
No	No	Yes	Yes	No	`UNFMT; HEX ON`
No	No	Yes	No	Yes	`HEXDUMP`

Displaying Records of Specific Record Type(s)

Of the typed records selected for display by any FROM, FOR and/or FILTER options, those **not** assigned specific record-type(s) may be temporarily suppressed from the display using the VIEW operation. By default, all records are displayed whether or not they are typed.

The VIEW operation may be invoked as follows:

1. On intial load using the VIEW parameter of the EDIT or BROWSE primary commands.
2. Executing VIEW primary command.
3. Executing the prefix (line) command V, V+ or V- against a record of the required record-type.
4. By positioning the cursor an a record (or shadow line) of the required record type and selecting one of the following options from the SDE Edit/Browse Utility Menu (assigned to <PF4>):

    - "1. VIEW *record_type* records only" (assigned to <PF1>)
    - "2. Add/Remove *record_type* records to/from current VIEW" (assigned to <PF2>)
    - "4. Select from list of available record-types" (assigned to <PF4>)

VIEW nominates one or more record-types which identify those typed records to be made visible in the display. Alternatively, records-types may be added or suppressed from the existing display of visible records using VIEW + (plus) or VIEW - (minus) respectively. e.g. To display only records assigned record-type "REC-CUST" or "REC-ORDER", execute the following from the SDE window view command line:

```
VIEW  REC-CUST, REC-ORDER
```

To then add suppressed records assigned record-type "REC-PAYMENT" to this display, execute the following:

```
VIEW  + REC-PAYMENT
```

To re-display **all** records, execute the following:

```
VIEW  *
```

Unmapped (Untyped) records have internal record-type "UnMapped" whereas unmapped secondary segments have internal record-type "UnMappedSeg." If these unmapped records or segments are not selected for display by a VIEW operation, they are flagged as being "NOT SELECTED". If SHADOW NOTSELECTED is set on (the default), then consecutive occurrences of these records are represented by a single shadow line of type "NOT SELECTED".

Typed records or segments that are not selected for display by a VIEW operation are flagged as being SUPPRESSED. If SHADOW SUPPRESSED is set on (the default), consecutive suppressed records of the same record type are represented by a single shadow line of type "SUPPRESSED".

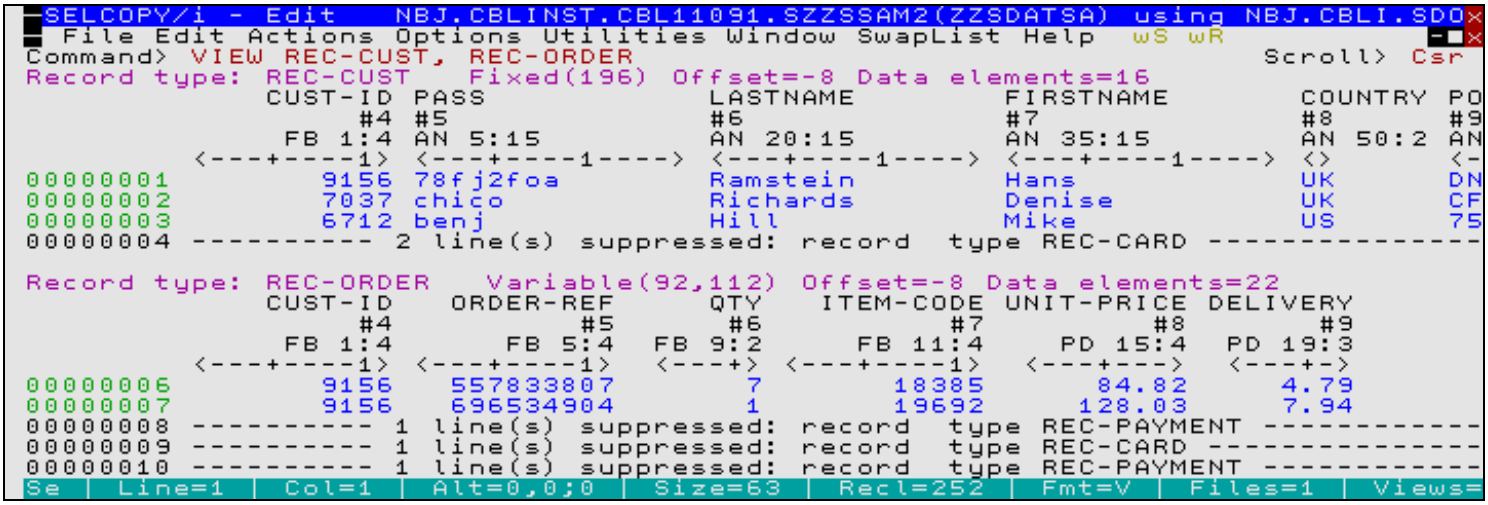

SELCOPY/i - Edit NBJ.CBLINST.CBL11091.SZZSSAM2(ZZSDATSA) using NBJ.CBLI.SDO
File Edit Actions Options Utilities Window SwapList Help wS wR
Command> VIEW REC-CUST, REC-ORDER
Scroll> Csr
Record type: REC-CUST Fixed(196) Offset=-8 Data elements=16
CUST-ID PASS LASTNAME FIRSTNAME COUNTRY PO
#4 #5 #6 #7 #8 #9
FB 1:4 AN 5:15 AN 20:15 AN 35:15 AN 50:2 AN
00000001 9156 78fj2foa Ramstein Hans UK DN
00000002 7037 chico Richards Denise UK CF
00000003 6712 benj Hill Mike US 75
00000004 ---------- 2 line(s) suppressed: record type REC-CARD ----------
Record type: REC-ORDER Variable(92,112) Offset=-8 Data elements=22
CUST-ID ORDER-REF QTY ITEM-CODE UNIT-PRICE DELIVERY
#4 #5 #6 #7 #8 #9
FB 1:4 FB 5:4 FB 9:2 FB 11:4 PD 15:4 PD 19:3
00000006 9156 557833807 7 18385 84.82 4.79
00000007 9156 696534904 1 19692 128.03 7.94
00000008 ---------- 1 line(s) suppressed: record type REC-PAYMENT ----------
00000009 ---------- 1 line(s) suppressed: record type REC-CARD ----------
00000010 ---------- 1 line(s) suppressed: record type REC-PAYMENT ----------
Se | Line=1 | Col=1 | Alt=0,0;0 | Size=63 | Recl=252 | Fmt=V | Files=1 | Views=

Figure 14. SDE View Selected Record Types.

Display Unformatted Record Data

The initial format of the record data in the SDE window view is determined by the FORMAT parameter of the EDIT or BROWSE command (default is FORMAT TABLE).

If a structure has been specified, on the SDE Data Edit panel or via the EDIT/BROWSE USING parameter, then record type assignment occurs so that each record is Typed.

Furthermore, if FORMAT TABLE or FORMAT SINGLE is in effect, the records are formatted so that record data is arranged and displayed in distinct, discrete fields. FORMAT TABLE displays records, beginning at the first selected record, in multi record view, whereas FORMAT single displays the first selected record in single record view.

Once the SDE window view has been opened, the display of data may be changed so that it is unformatted and optionally untyped.

To display records as unformatted but preserve their record-type assignment (so continuing the ability to selectively VIEW only records assigned specific record-types), then execute the FORMAT command with parameter CHAR or HEX. Note that the record view (multi or single) is unchanged.

To display records as unformatted and untyped, then execute UNFMT to display the data in single record view or CHAR to display the data in multi record view. To further display both character and hex representation of the data, use HEX ON/OFF.

Unformatted record data is displayed as a single character field of length equal to the record length and with field name "UnMapped" or "UnMappedSeg".

Notes:

1. FORMAT HEX is equivalent to FORMAT CHAR with HEX set on.
2. If the record data contains non-printable characters, then the CHAR representation of the data is non-enterable. This protects the user from accidental update of the non-printable text that would otherwise occur as a result of 3270 I/O. Update of record data may still be achieved by updating the HEX representation of the text or using the CHANGE command.

Displaying a Record/Segment in Single Record View

A single record view arranges the fields vertically. If formatted, the field names and formatted data types is displayed to the left of the associated field data.

Once the SDE window view has been opened, a single record view of a data record may be displayed using any of the following:

- Where the current display is a multi record view and current record-type, formatting and hex display is to be preserved, execute the ZOOM operation as follows:

    1. Enter primary command ZOOM SDE window command prompt, then position the cursor on the required record before pressing <Enter>.
    2. Enter prefix (line) command "Z" against the required record.
    3. Press <PF2> with the cursor positioned on the required record. As distributed, <PF2> is assigned to the macro, SDEZOOMW, which opens a new window to display the single record view.
    4. Position the cursor on the required, press <PF4> to open the *"SDE Browse/Edit Utilities Menu"* or *"DB2 Browse/Edit Utilities Menu"* as appropriate, then select option *"Open single-record (ZOOM) view in new window"* to invoke the SDEZOOMW macro.

- To display formatted data using the currently assigned record-type, execute FORMAT SINGLE with the cursor positioned on the required record.
- To re-assign the record-type and display formatted data, execute MAP with the cursor positioned on the required record or execute prefix (line) command "MAP" against the required record when in multi record view.
- To display unformatted data, execute UNFMT with the cursor positioned on the required record.

Only records that are visible may be viewed in the single record view (i.e. records that are flagged as being NOTSELECTED, SUPPRESSED or EXCLUDED cannot be viewed in this format.)

When in single record view, commands LEFT and RIGHT (<PF10> and <PF11> respectively), and/or NEXT and PREV scroll to other visible records or record segments.

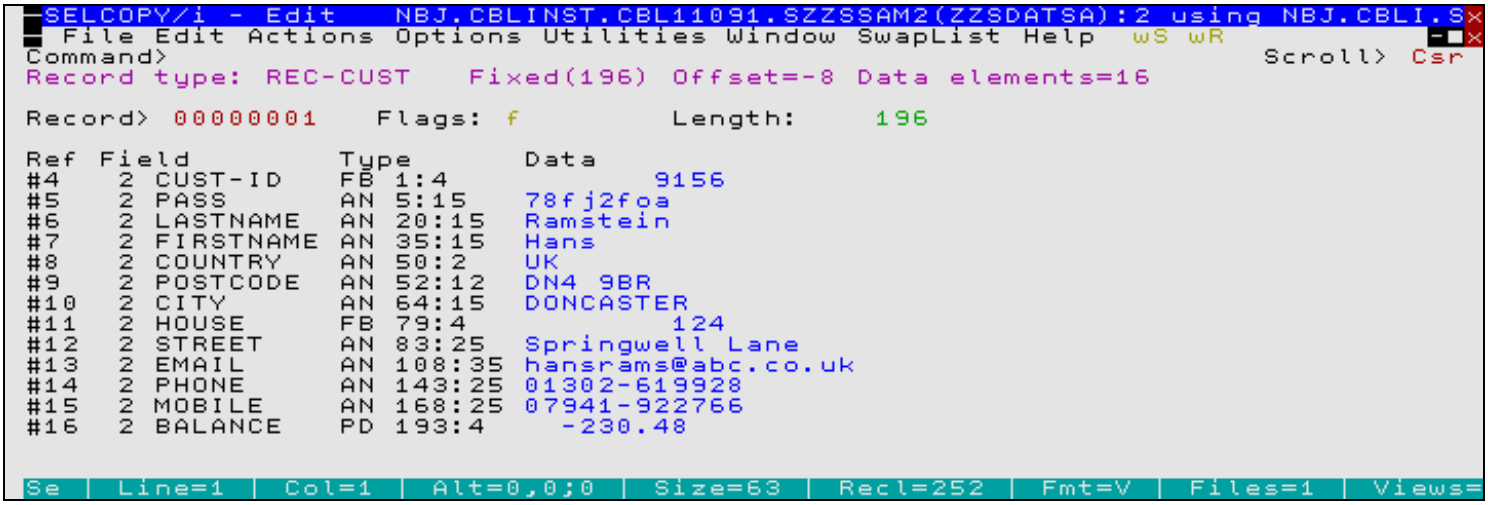

SELCOPY/i - Edit NBJ.CBLINST.CBL11091.SZZSSAM2(ZZSDATSA):2 using NBJ.CBLI.S
File Edit Actions Options Utilities Window SwapList Help wS wR
Command>
Scroll> Csr
Record type: REC-CUST Fixed(196) Offset=-8 Data elements=16
Record> 00000001 Flags: f Length: 196
Ref Field Type Data
#4 2 CUST-ID FB 1:4 9156
#5 2 PASS AN 5:15 78fj2foa
#6 2 LASTNAME AN 20:15 Ramstein
#7 2 FIRSTNAME AN 35:15 Hans
#8 2 COUNTRY AN 50:2 UK
#9 2 POSTCODE AN 52:12 DN4 9BR
#10 2 CITY AN 64:15 DONCASTER
#11 2 HOUSE FB 79:4 124
#12 2 STREET AN 83:25 Springwell Lane
#13 2 EMAIL AN 108:35 hansrams@abc.co.uk
#14 2 PHONE AN 143:25 01302-619928
#15 2 MOBILE AN 168:25 07941-922766
#16 2 BALANCE PD 193:4 -230.48
Se | Line=1 | Col=1 | Alt=0,0;0 | Size=63 | Recl=252 | Fmt=V | Files=1 | Views=

Figure 15. SDE Formatted Single Record View.

Notes:

1. An unformatted record in single record view is displayed with a single field name of "UnMapped" or "UnMappedSeg" and all record data displayed to the right of the field name.
2. Executing ZOOM in a FORMAT TABLE view is equivalent to executing the FORMAT SINGLE command.
3. Executing ZOOM when already in single record view, will switch back to a multiple record view preserving current record-type, formatting and hex display.

Displaying Hexadecimal Representation of Record Data

FORMAT HEX displays the unformatted record's data in both character and up/down hexadecimal.

Alternatively, the character and up/down hexadecimal representation may be applied to all fields within a formatted or unformatted record using HEX ON/OFF.

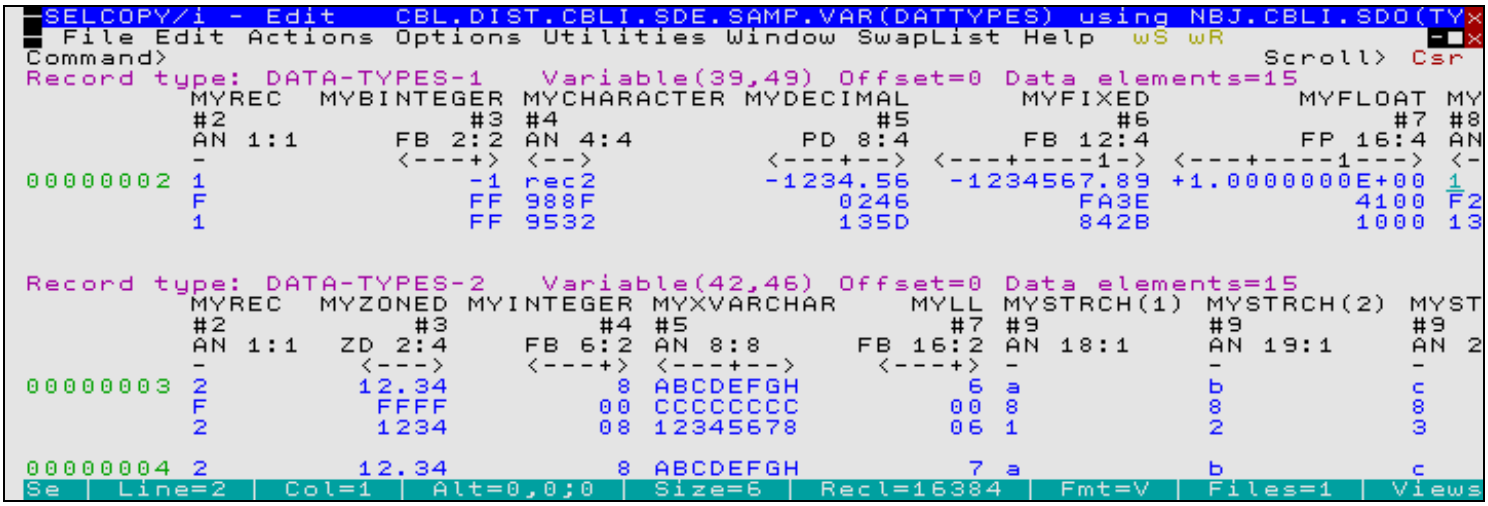

SELCOPY/i - Edit CBL.DIST.CBLI.SDE.SAMP.VAR(DATTYPES) using NBJ.CBLI.SDO(TY
File Edit Actions Options Utilities Window SwapList Help wS wR
Command>
Scroll> Csr
Record type: DATA-TYPES-1 Variable(39,49) Offset=0 Data elements=15
MYREC MYBINTEGER MYCHARACTER MYDECIMAL MYFIXED MYFLOAT MY
#2 #3 #4 #5 #6 #7 #8
AN 1:1 FB 2:2 AN 4:4 PD 8:4 FB 12:4 FP 16:4 AN
00000002 1 -1 rec2 -1234.56 -1234567.89 +1.0000000E+00 1
F FF 988F 0246 FA3E 4100 F2
1 FF 9532 135D 842B 1000 13
Record type: DATA-TYPES-2 Variable(42,46) Offset=0 Data elements=15
MYREC MYZONED MYINTEGER MYXVARCHAR MYLL MYSTRCH(1) MYSTRCH(2) MYST
#2 #3 #4 #5 #7 #9 #9 #9
AN 1:1 ZD 2:4 FB 6:2 AN 8:8 FB 16:2 AN 18:1 AN 19:1 AN 2
00000003 2 12.34 8 ABCDEFGH 6 a b c
F FFFF 00 CCCCCCCC 00 8 8 8
2 1234 08 12345678 06 1 2 3
00000004 2 12.34 8 ABCDEFGH 7 a b c
Se | Line=2 | Col=1 | Alt=0,0;0 | Size=6 | Recl=16384 | Fmt=V | Files=1 | Views

Figure 16. SDE Formatted Multiple Record View with HEX ON.

Excluding Records from the Display

By default, the contents of records that are flagged as NOT SELECTED or SUPPRESSED are not displayed in the SDE window view.

Further filtering of the displayed records can be achieved by setting the EXCLUDED flag on those records that are to be excluded from the display. If SHADOW EXCLUDED is set on (the default), consecutive excluded records that are assigned the same record type are represented by a single shadow line of type "EXCLUDED".

The Prefix Area Command X(n) or XX may be used to exclude a record (and optionally n-1 records that follow) or a block of records, regardless of assigned record type.

Alternatively, the EXCLUDE command may be used to exclude records of a specific record type that fall within a range of records and also match a supplied search string. Records that are assigned a different record type are unaffected. EXCLUDE is similar to the ISPF style text edit command with additional syntax to nominate those columns to be searched.

e.g. Exclude all records assigned the same record type as the focus record and contain the character string "London" (any case) within either of the fields named "X_City" or "X_County".

```
EXCLUDE  ALL  'London'  (X_City, X_County)
```

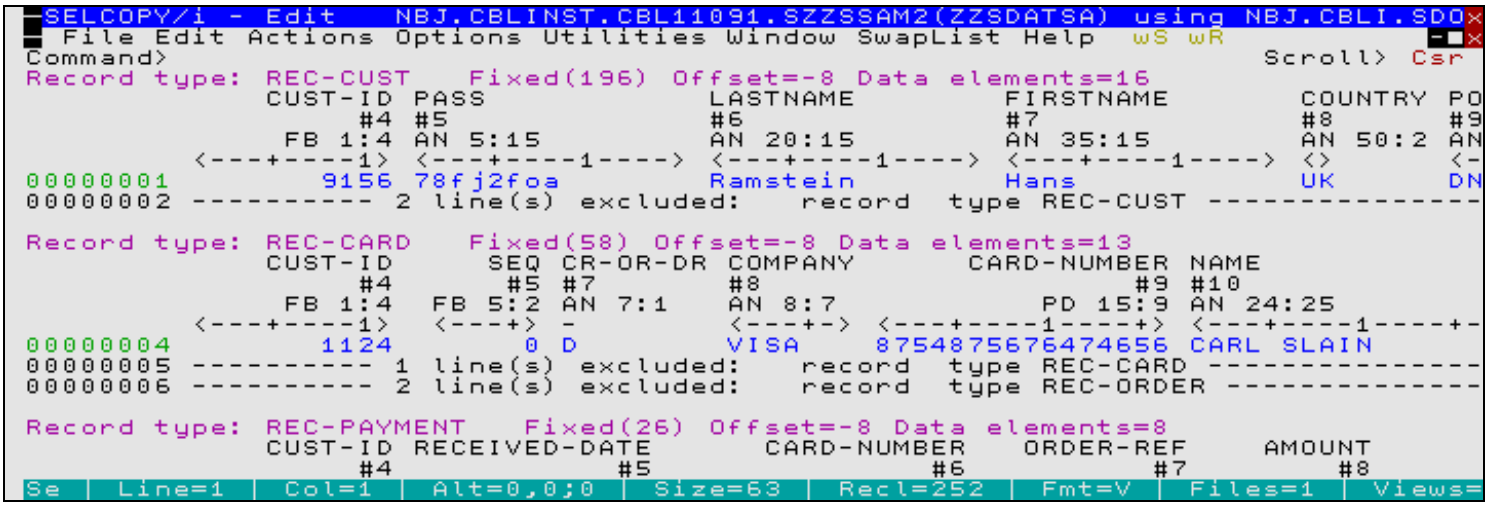

SELCOPY/i - Edit NBJ.CBLINST.CBL11091.SZZSSAM2(ZZSDATSA) using NBJ.CBLI.SDO
File Edit Actions Options Utilities Window SwapList Help wS wR
Command>
Scroll> Csr
Record type: REC-CUST Fixed(196) Offset=-8 Data elements=16
CUST-ID PASS LASTNAME FIRSTNAME COUNTRY PO
#4 #5 #6 #7 #8 #9
FB 1:4 AN 5:15 AN 20:15 AN 35:15 AN 50:2 AN
00000001 9156 78fj2foa Ramstein Hans UK DN
00000002 ---------- 2 line(s) excluded: record type REC-CUST ----------
Record type: REC-CARD Fixed(58) Offset=-8 Data elements=13
CUST-ID SEQ CR-OR-DR COMPANY CARD-NUMBER NAME
#4 #5 #7 #8 #9 #10
FB 1:4 FB 5:2 AN 7:1 AN 8:7 PD 15:9 AN 24:25
00000004 1124 0 D VISA 8754875676474656 CARL SLAIN
00000005 ---------- 1 line(s) excluded: record type REC-CARD ----------
00000006 ---------- 2 line(s) excluded: record type REC-ORDER ----------
Record type: REC-PAYMENT Fixed(26) Offset=-8 Data elements=8
CUST-ID RECEIVED-DATE CARD-NUMBER ORDER-REF AMOUNT
#4 #5 #6 #7 #8
Se | Line=1 | Col=1 | Alt=0,0;0 | Size=63 | Recl=252 | Fmt=V | Files=1 | Views=

Figure 17. SDE Excluded Records.

The FLIP command will flip the EXCLUDED flag on all records of a specific record type so that visible records are excluded and excluded records are brought back into view.

Excluded records may also be brought back into view using the Prefix Area Commands, F(n) and L(n). These remove the EXCLUDED flag from the First or Last records respectively (and optionally n-1 records that follow/precede) that belong to a block of excluded records, regardless of record type.

Alternatively, the RESET EXCLUDED command may be executed to remove the EXCLUDED flag from All excluded records of a specific record type.

SDE commands that perform searches and selection on records may also alter the EXCLUDED flag setting of individual records in the display. See *"Searching Data Records"* and commands FIND, LOCATE, WHERE, MORE and LESS.

Displaying Selected Columns

By default, all fields belonging to formatted records in the SDE window view are displayed in the order in which they occur within the data records.

Once the SDE window view has been opened, the SELECT command may be used to re-order and/or restrict the number of fields displayed in records that are assigned the specified record type. In addition to this, the HOLD parameter may be used to indicate that a specified field, and all fields before it, are to be held when scrolling the display of the record data left and right.

The supplied macro, SDESEL, may be executed to generate a SELECT command in a temporary CBLe text edit view referencing all fields in the focus record group or all record-types within the SDO.

This is particularly useful when dealing with record types with a large number of fields required for display. Using standard text editing techniques, the user can manipulate the SELECT command to tailor the presence and order of fields within the SELECT command without having to re-type the field names or references.

Once configured, the command may be executed by positioning the cursor on the first line of the command and pressing <PF4> to invoke the CMDTEXT facility. The SDE window view behind the text edit view is updated.

SDESEL may be invoked as follows:

- Position the cursor on a record assigned the required record-type and press <PF14>.
- Position the cursor on a record assigned the required record-type and press <PF4> to open the *"SDE Browse/Edit Utilities Menu"* or *"DB2 Browse/Edit Utilities Menu"* as appropriate, then select option *"Select/Exclude/Order visible field-names"* .

e.g. To re-order the occurrence of fields in records of record type "REC-CUST" and hold the first two fields, execute the following. Note that "*" (asterisk) represents all remaining fields that have **not** already been specified on the SELECT command, in the order in which they occur within the record.

```
SELECT   FIRSTNAME, LASTNAME HOLD, EMAIL, *    FROM REC-CUST
```

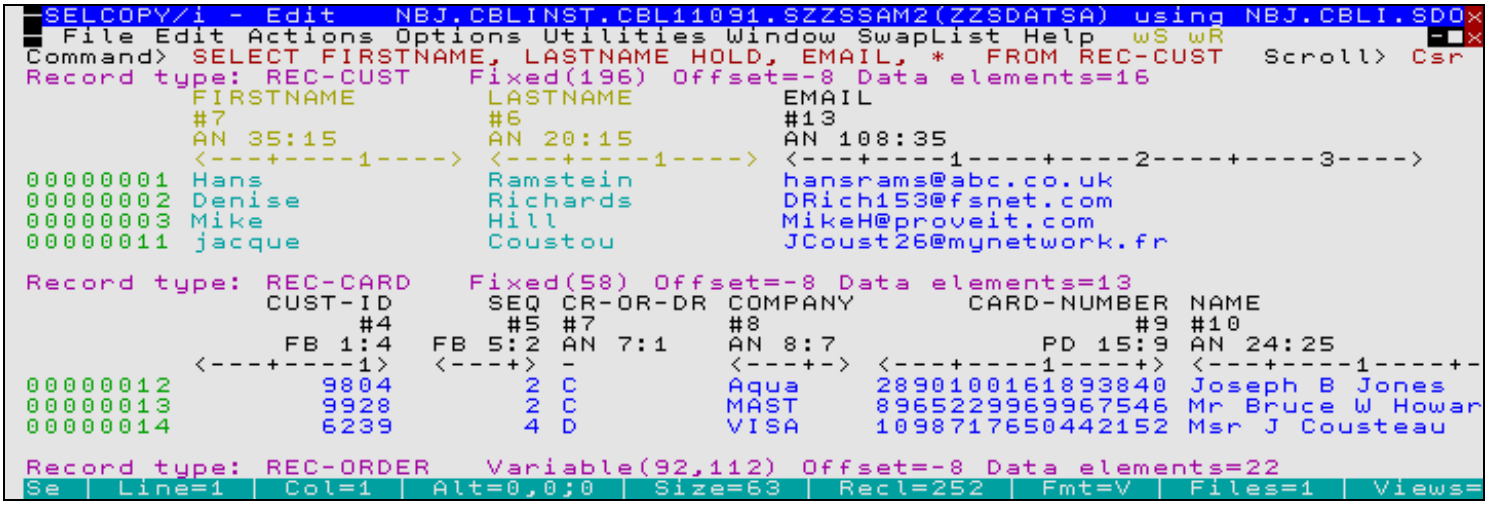

SELCOPY/i - Edit NBJ.CBLINST.CBL11091.SZZSSAM2(ZZSDATSA) using NBJ.CBLI.SDO
File Edit Actions Options Utilities Window SwapList Help wS wR
Command> SELECT FIRSTNAME, LASTNAME HOLD, EMAIL, * FROM REC-CUST Scroll> Csr
Record type: REC-CUST Fixed(196) Offset=-8 Data elements=16
FIRSTNAME LASTNAME EMAIL
#7 #6 #13
AN 35:15 AN 20:15 AN 108:35
00000001 Hans Ramstein hansrams@abc.co.uk
00000002 Denise Richards DRich153@fsnet.com
00000003 Mike Hill MikeH@proveit.com
00000011 jacque Coustou JCoust26@mynetwork.fr
Record type: REC-CARD Fixed(58) Offset=-8 Data elements=13
CUST-ID SEQ CR-OR-DR COMPANY CARD-NUMBER NAME
#4 #5 #7 #8 #9 #10
FB 1:4 FB 5:2 AN 7:1 AN 8:7 PD 15:9 AN 24:25
00000012 9804 2 C Aqua 2890100161893840 Joseph B Jones
00000013 9928 2 C MAST 8965229969967546 Mr Bruce W Howar
00000014 6239 4 D VISA 1098717650442152 Msr J Cousteau
Record type: REC-ORDER Variable(92,112) Offset=-8 Data elements=22
Se | Line=1 | Col=1 | Alt=0,0;0 | Size=63 | Recl=252 | Fmt=V | Files=1 | Views=

Figure 18. SDE Selected Field Columns.

By default, fields that are not assigned a field name (e.g. COBOL FILLER fields) are not displayed. In order to display these fields, use SET UNNAMED ON.

Displaying the Record Headers

In a multi record view of formatted data, then all 5 field column header lines are displayed by default before each group of records of the same record type.

These are the **Record Type**, **Field Name**, **Field Reference**, **Field Type** and **Scale** header lines which are described in detail under *"SDE Window Views"*.

The **Field Reference**, **Field Type** and **Scale** lines may be set on or off using the SET REFERENCE, SET TYPE and SET SCALE commands respectively.

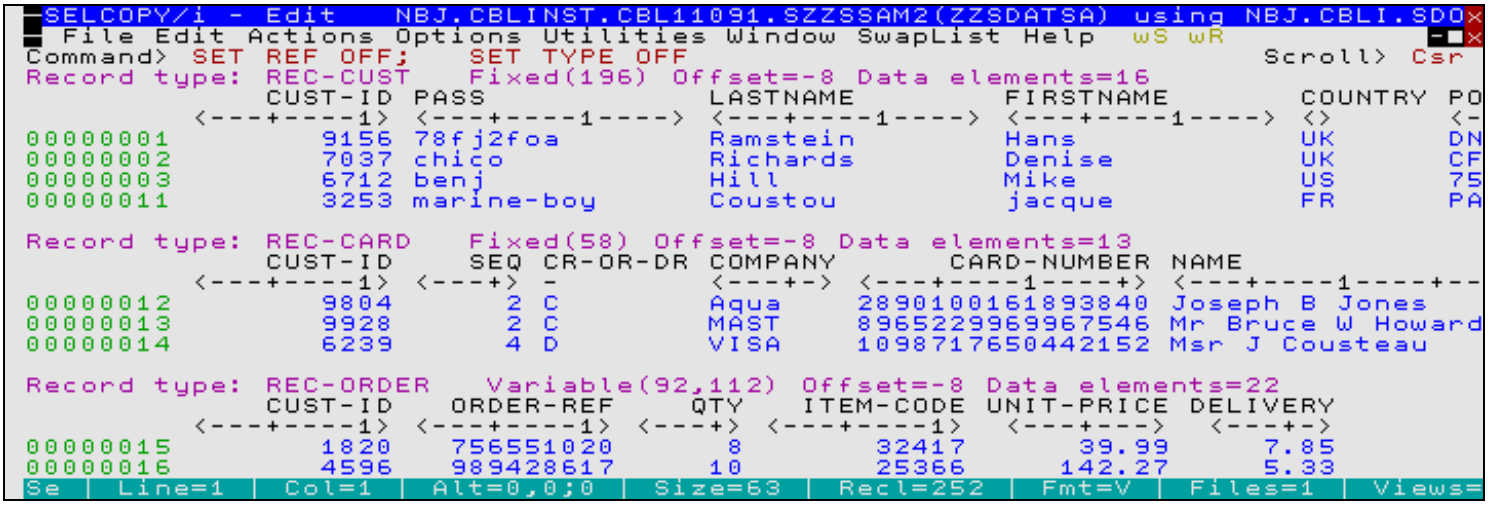

SELCOPY/i - Edit NBJ.CBLINST.CBL11091.SZZSSAM2(ZZSDATSA) using NBJ.CBLI.SDO
File Edit Actions Options Utilities Window SwapList Help wS wR
Command> SET REF OFF; SET TYPE OFF Scroll> Csr
Record type: REC-CUST Fixed(196) Offset=-8 Data elements=16
CUST-ID PASS LASTNAME FIRSTNAME COUNTRY PO
00000001 9156 78fj2foa Ramstein Hans UK DN
00000002 7037 chico Richards Denise UK CF
00000003 6712 benj Hill Mike US 75
00000011 3253 marine-boy Coustou jacque FR PA
Record type: REC-CARD Fixed(58) Offset=-8 Data elements=13
CUST-ID SEQ CR-OR-DR COMPANY CARD-NUMBER NAME
00000012 9804 2 C Aqua 2890100161893840 Joseph B Jones
00000013 9928 2 C MAST 8965229969967546 Mr Bruce W Howard
00000014 6239 4 D VISA 1098717650442152 Msr J Cousteau
Record type: REC-ORDER Variable(92,112) Offset=-8 Data elements=22
CUST-ID ORDER-REF QTY ITEM-CODE UNIT-PRICE DELIVERY
00000015 1820 756551020 8 32417 39.99 7.85
00000016 4596 989428617 10 25366 142.27 5.33
Se Line=1 Col=1 Alt=0,0;0 Size=63 Recl=252 Fmt=V Files=1 Views=

Figure 19. SDE Suppressed Header Lines.

Searching Data Records

SDE supports powerful search commands that allow the user to find and create subsets of records based on specified search criteria. (FIND, LOCATE, WHERE, MORE and LESS)

These are covered by the following:

- Finding Record Data by Search String
- Locating Records Using a Search Expression
- Creating a Subset of Records for Display

Finding Record Data by Search String

Like EXCLUDE (see *"Excluding Records from the Display"*), the FIND and RFIND commands are similar to the ISPF style text edit commands with additional syntax to nominate those columns to be searched.

For formatted records, the search scope for FIND is supplied as a list of individual fields and/or field ranges that occur in records that match a specific record type. Records that are assigned a different record type and fields that have been removed from view by a SELECT command, are automatically excluded from the search.

e.g. To find the first occurrence of the mixed case string 'James' within the character fields "LASTNAME" or "FIRSTNAME" (field reference #4) which belong to records of the same record type as the focus record, then enter the following from the SDE command prompt:

```
FIND  FIRST  C'James'   (LASTNAME, #4)
```

Un-quoted, numeric search strings are treated as numeric values when testing non-character data fields. i.e. the value of the numeric field data must be arithmetically equal to the search value. When testing character fields, numeric search strings are treated as character data.

e.g. To find all occurrences of the value "1" within any numeric field or the character string "1" in any character field within a record of the same record type as the focus record, then enter the following from the SDE command prompt:

```
FIND  ALL  1   #ALL
```

If the specified occurrence (ALL, FIRST, LAST, NEXT or PREV) of the search string or numeric value is found, then:

1. If the record is EXCLUDED, the EXCLUDED flag is set off so that the record is brought back into view.
2. The cursor is positioned at the beginning of the string or numeric field.
3. If necessary, scrolling occurs to display the found data.

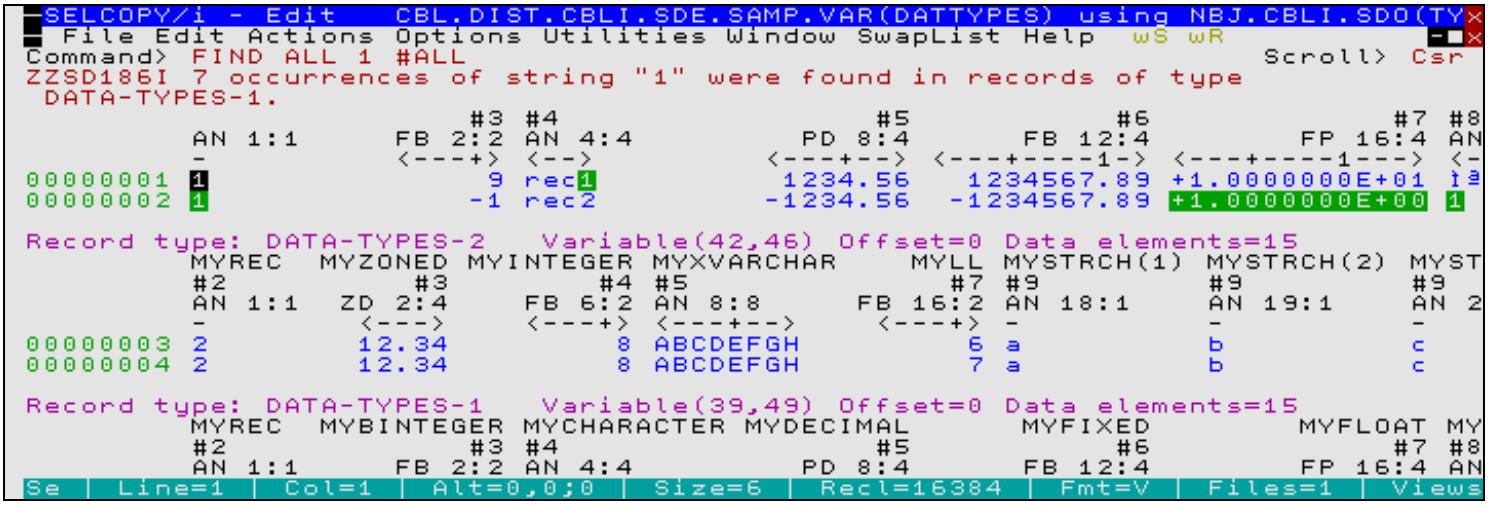

SELCOPY/i - Edit CBL.DIST.CBLI.SDE.SAMP.VAR(DATTYPES) using NBJ.CBLI.SDO(TY
File Edit Actions Options Utilities Window SwapList Help wS wR
Command> FIND ALL 1 #ALL
Scroll> Csr
ZZSD186I 7 occurrences of string "1" were found in records of type
DATA-TYPES-1.
00000001 1 9 rec1 1234.56 1234567.89 +1.0000000E+01
00000002 1 -1 rec2 -1234.56 -1234567.89 +1.0000000E+00 1
Record type: DATA-TYPES-2 Variable(42,46) Offset=0 Data elements=15
00000003 2 12.34 8 ABCDEFGH 6 a b c
00000004 2 12.34 8 ABCDEFGH 7 a b c
Record type: DATA-TYPES-1 Variable(39,49) Offset=0 Data elements=15
Se Line=1 Col=1 Alt=0,0;0 Size=6 Recl=16384 Fmt=V Files=1 Views

Figure 20. SDE FIND Value.

Locating Records Using a Search Expression

The LOCATE command will scroll the display to make a specific or relative record number the current (first visible) line of the window view. Alternatively, a record of a specific record type that matches the search criteria is made the current line of the display.

LOCATE supports use of an SDE expression which allows for more sophisticated search criteria than that provided by FIND.

e.g. Locate the last record with the same record type as the focus record, where the field "UNIT-PRICE" has a value greater than 100 **and either** the field "DELIVERY" has a value less than 10 **or** field "REGION" is "UK".

```
LOCATE  LAST  UNIT-PRICE > 100  &  ( DELIVERY < 10  |  REGION = 'UK' )
```

Creating a Subset of Records for Display

SDE expressions may be used as filters to display only a subset of records, record segments or DB2 table rows that satisfy the expression criteria.

The WHERE command filters all records of the default record type so that only those records that match the supplied expression criteria are displayed. Records that **do not** match the expression are excluded whereas records that are assigned a different record-type are unaffected.

e.g. Display only those records assigned the same record-type as the focus record and where the contents of field "DISPATCH_DATE" is less than the contents of field "ORDER_DATE" **or** either of the following are **not** true:

1. Value of numeric field "ORDER_VALUE" is greater than or equal to 256.55.
2. Value of numeric field "ORDER_BALANCE" is equal to 0.

```
WHERE  (DISPATCH_DATE < ORDER_DATE)  |  \(ORDER_VALUE >= 256.55  &  ORDER_BALANCE = 0)
```

The LESS command allows the user to filter the subset further so that only non-excluded records of the default record type are excluded if they **do not** satisfy the supplied expression criteria. Records that are already excluded or assigned a different record-type are unaffected.

e.g. Exclude only records that are in view that are assigned the same record-type as the focus record and where the contents of field "ORDER_DATE" is greater than or equal to '2007/11/28'.

```
LESS    ORDER_DATE >= '2007/11/28'
```

Similarly, the MORE command allows the user to add to the subset of records so that excluded records of the default record type are made visible if they satisfy the supplied expression criteria. Records that are already visible or assigned a different record-type are unaffected.

e.g. Bring back into view only excluded records that are assigned the same record-type as the focus record and where the character field "BACKUP_VOL" **does not** begin with the string "Z16".

```
MORE    BACKUP_VOL \>> C'Z16'
```

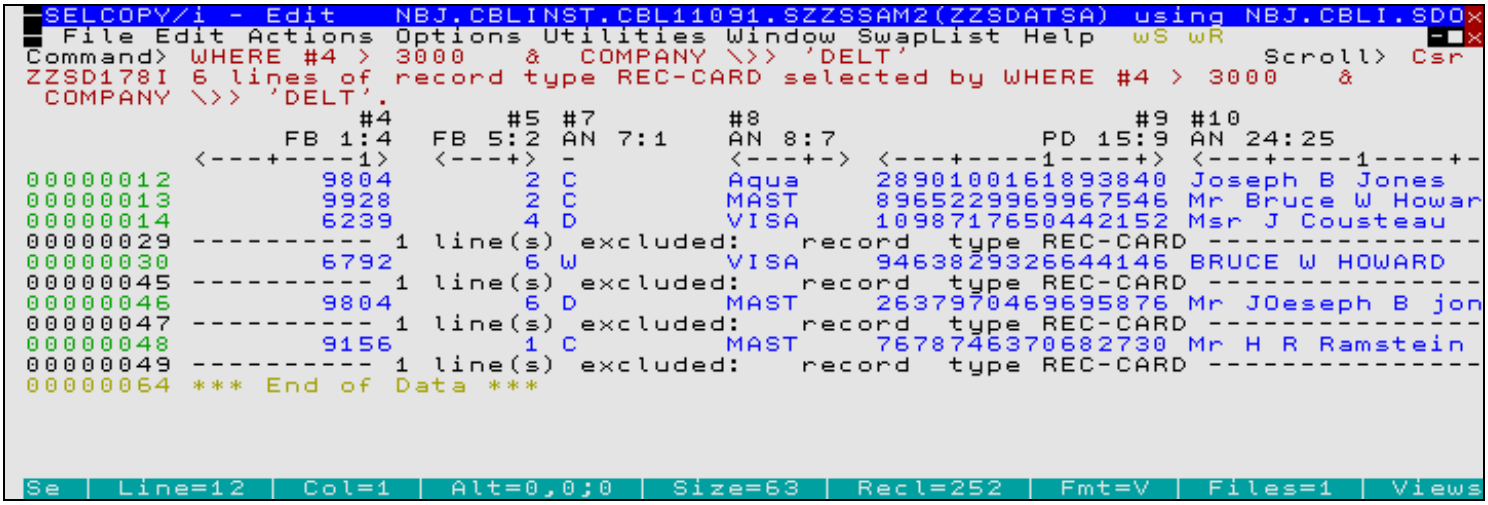

SELCOPY/i - Edit NBJ.CBLINST.CBL11091.SZZSSAM2(ZZSDATSA) using NBJ.CBLI.SDO
File Edit Actions Options Utilities Window SwapList Help wS wR
Command> WHERE #4 > 3000 & COMPANY \>> 'DELT' Scroll> Csr
ZZSD178I 6 lines of record type REC-CARD selected by WHERE #4 > 3000 &
COMPANY \>> 'DELT'.
00000012 9804 2 C Aqua 2890100161893840 Joseph B Jones
00000013 9928 2 C MAST 8965229969967546 Mr Bruce W Howar
00000014 6239 4 D VISA 1098717650442152 Msr J Cousteau
00000029 ---------- 1 line(s) excluded: record type REC-CARD ----------
00000030 6792 6 W VISA 9463829326644146 BRUCE W HOWARD
00000045 ---------- 1 line(s) excluded: record type REC-CARD ----------
00000046 9804 6 D MAST 2637970469695876 Mr JOeseph B jon
00000047 ---------- 1 line(s) excluded: record type REC-CARD ----------
00000048 9156 1 C MAST 7678746370682730 Mr H R Ramstein
00000049 ---------- 1 line(s) excluded: record type REC-CARD ----------
00000064 *** End of Data ***
Se | Line=12 | Col=1 | Alt=0,0;0 | Size=63 | Recl=252 | Fmt=V | Files=1 | Views

Figure 21. SDE Subset Using WHERE.

Editing Data

SELCOPY/i structured data editor supports standard functions required for basic text edit but with additional emphasis given to recognition, validation and preservation of field data length and data type.

Basic editing features include:

- Typing New Record Data
- Find and Replace Data
- Inserting Records
- Deleting Records
- Moving Records
- Copying Records
- Duplicating Records
- Undo and Redo Changes

In order to perform edit tasks, the SDE window view must have been opened for EDIT, not BROWSE. In addition, the edit technique employed to edit the data also governs which of these editing features are supported and when data may be saved.

Changes to edited text are saved in storage. Unless QQUIT is used, the user will be prompted to save the changes to disk when the file is closed. Depending on the edit technique being used, the SAVE command should be used to save the changes without closing the file.

Typing New Record Data

Simply position the cursor within the record at which text is to be entered and begin typing. Existing data may be overtyped and deleted, or new text may be inserted within the existing record data.

Note that the length of a record, record segment or DB2 table row cannot be updated simply by inserting or deleting data. Record length changes may be achieved by manually updating the Record information "Length" field (See command RECLENGTH) or via a CHANGE operation.

Beware when editing typed, unformatted data that the integrity of the data when formatted is preserved. i.e. Inserting, deleting or overtyping data so that, when formatted, unintended or invalid data exists within the formatted fields.

When editing formatted data, text entered within a field is restricted by the size of that field. On encountering the end of a field, the cursor is automatically positioned at the next field in the display. Therefore, users should beware not to accidentally overtype data in subsequent fields. Note, however, that inserting text before existing data in a field will not propagate that data into subsequent fields in the display.

If character field data contains non-printable text, then areas of the field containing printable text are highlighted (using underscore characters by default) and areas of non-printable text are fixed within the field data. Only areas of printable text are enterable. However, if HEX ON or FORMAT HEX is in effect, then non-printable text may be updated by overtyping the hexadecimal representation of the data.

On inserting data in the variable length character field of a formatted record, the new length of the field data will automatically be updated in the accompanying length field. Similarly, any alteration to the value in the length field will automatically be reflected in the variable length character data (i.e. the data will be truncated or padded with blanks.)

On entering data in formatted numeric fields, the following occurs:

1. The field is scanned from left to right and leading blanks ignored.
2. The numeric data is identified as being the non-blank text terminated by the end of the field or a blank character.
3. Data within the field that is to the right of a terminating blank is ignored.
4. The numeric data is interpreted and validated to establish whether it complies with the field's defined data type, precision and scale.

If an invalid value is entered in a numeric field then a warning popup window is opened, giving the reason for rejecting the value and prompting the user to either enter a valid numeric value or discard the change for that particular field. This occurs for each field that has been updated with an invalid numeric value since the last transaction.

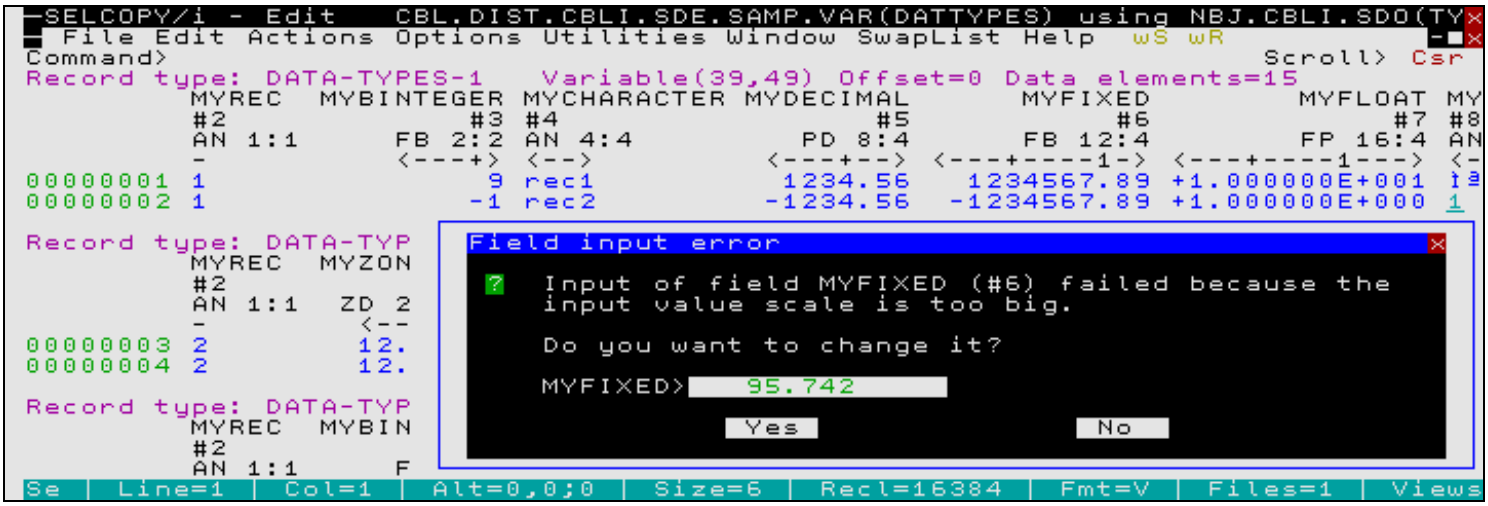

SELCOPY/i - Edit CBL.DIST.CBLI.SDE.SAMP.VAR(DATTYPES) using NBJ.CBLI.SDO(TY
File Edit Actions Options Utilities Window SwapList Help wS wR
Command>
Scroll> Csr
Record type: DATA-TYPES-1 Variable(39,49) Offset=0 Data elements=15
MYREC MYBINTEGER MYCHARACTER MYDECIMAL MYFIXED MYFLOAT MY
#2 #3 #4 #5 #6 #7 #8
AN 1:1 FB 2:2 AN 4:4 PD 8:4 FB 12:4 FP 16:4 AN
00000001 1 9 rec1 1234.56 1234567.89 +1.000000E+001
00000002 1 -1 rec2 -1234.56 -1234567.89 +1.000000E+000 1
Record type: DATA-TYP
MYREC MYZON
#2
AN 1:1 ZD 2
00000003 2 12.
00000004 2 12.
Record type: DATA-TYP
MYREC MYBIN
#2
AN 1:1 F
Field input error
Input of field MYFIXED (#6) failed because the input value scale is too big.
Do you want to change it?
MYFIXED> 95.742
Yes
No
Se | Line=1 | Col=1 | Alt=0,0;0 | Size=6 | Recl=16384 | Fmt=V | Files=1 | Views

Figure 22. SDE Field Input Error.

Find and Replace Data

The CHANGE command allows users to selectively change occurrences of a supplied character string or numeric value to another string/value, for records of a specific record type.

The CHANGE command will search for the first specified string/value using the same specifications used for a FIND command (see *"Finding Record Data by Search String"*), then replace it with the second specified string/value.

e.g. To find and change the next occurrence of the value "1234.56" to "9876.54" in the numeric field "MYDECIMAL", in records of the same record type as the focus record, enter the following:

```
CHANGE  1234.56  9876.54   (MYDECIMAL)
```

To find and change the next occurrence of the search string/value execute the RCHANGE SDE CLI command (assigned to PF6 by default). To find and optionally change the next occurrence of the search string/value execute a combination of RFIND (assigned to PF5 by default) and then RCHANGE.

CHANGE ALL will change all occurrences of the string/value in visible or EXCLUDED records of the default record type.

For character data, where the search and replace strings are of different length, then:

- The text following a found search string will be shifted left or right. If parameter TEXT has been specified, then, before shifting the text, blanks will inserted or absorbed as necessary to maintain the positions of blank delimited words that follow the replaced text.
- The length of data in a variable length field may be updated to be any value less than or equal to the maximum length of the field. Therefore, for unformatted data, the length of the record may be updated to any value less than or equal to the maximum record length.

Inserting Records

Record, record segment and DB2 table row insert is **not** supported by Update-in-place edit technique.

New lines of data (records or record segments of specific record-type or DB2 table rows) may be inserted before or after existing lines of data in the display as follows:

1. Enter prefix (line) command "I(n)" on the line after which the new records are to be inserted. The record-type assigned to the inserted lines will be the same as that assigned to the line in which the prefix command was entered.

2. Execute the INSERT primary command to insert one or more new lines of a specified record-type following the focus line. For formatted data, INSERT also supports specification of one or more field columns and the default values to be inserted in each field.

   e.g. Insert 3 new records of record type "DATA-TYPES-1" containing some initial values in fields "MYCHARACTER" (#4) , "MYDECIMAL" (#5) and "MYFLOAT" (#7).

```
INSERT  DATA-TYPES-1 (#4, #5, MYFLOAT)  VALUE("TEMP", 3569.93, 23.0e-12)  3
```

When inserting unformatted records/segments, the user is prompted to specify a valid length for the new record/segment after which the newly inserted unformatted line is intialised to blanks.

When inserting formatted records/segments/DB2 table rows, then, unless otherwise specified, fields are initialised as follow:

- Numeric fields are set to 0.
- Alpha-numeric (character) fields are set to blanks.

Following an insert, the cursor is placed on the first character of the first field of the first newly inserted line.

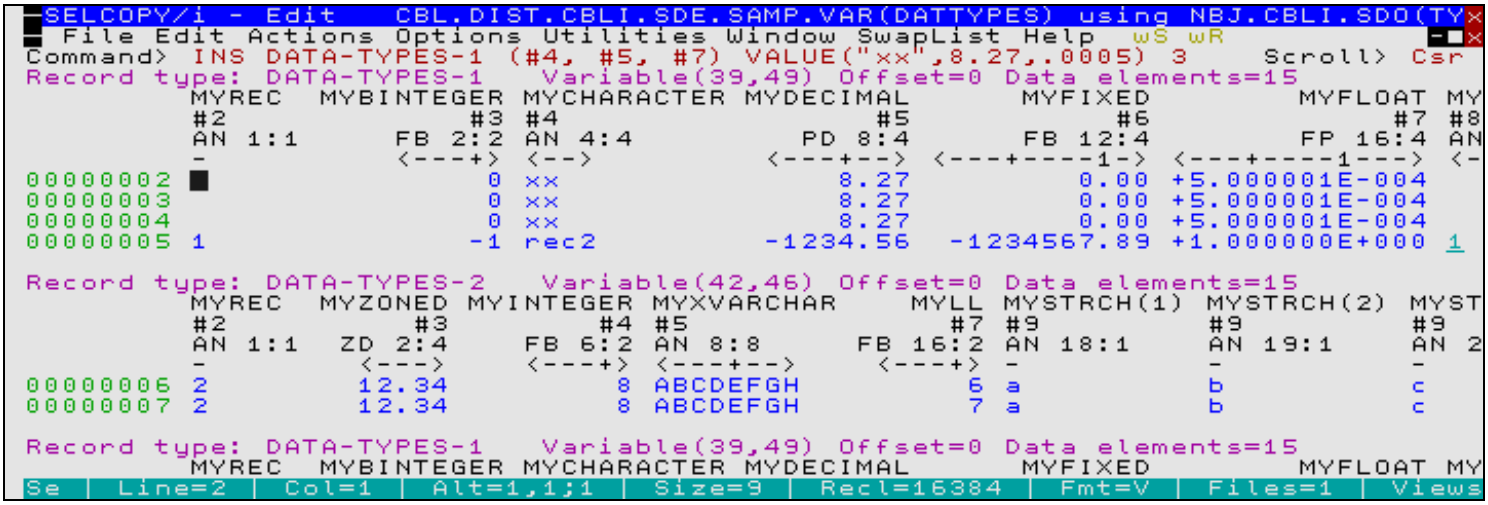

SELCOPY/i - Edit CBL.DIST.CBLI.SDE.SAMP.VAR(DATTYPES) using NBJ.CBLI.SDO(TY
File Edit Actions Options Utilities Window SwapList Help wS wR
Command> INS DATA-TYPES-1 (#4, #5, #7) VALUE("xx",8.27,.0005) 3 Scroll> Csr
Record type: DATA-TYPES-1 Variable(39,49) Offset=0 Data elements=15
MYREC MYBINTEGER MYCHARACTER MYDECIMAL MYFIXED MYFLOAT MY
#2 #3 #4 #5 #6 #7 #8
AN 1:1 FB 2:2 AN 4:4 PD 8:4 FB 12:4 FP 16:4 AN
00000002 0 xx 8.27 0.00 +5.000001E-004
00000003 0 xx 8.27 0.00 +5.000001E-004
00000004 0 xx 8.27 0.00 +5.000001E-004
00000005 1 -1 rec2 -1234.56 -1234567.89 +1.000000E+000 1
Record type: DATA-TYPES-2 Variable(42,46) Offset=0 Data elements=15
MYREC MYZONED MYINTEGER MYXVARCHAR MYLL MYSTRCH(1) MYSTRCH(2) MYST
#2 #3 #4 #5 #7 #9 #9 #9
AN 1:1 ZD 2:4 FB 6:2 AN 8:8 FB 16:2 AN 18:1 AN 19:1 AN 2
00000006 2 12.34 8 ABCDEFGH 6 a b c
00000007 2 12.34 8 ABCDEFGH 7 a b c
Record type: DATA-TYPES-1 Variable(39,49) Offset=0 Data elements=15
MYREC MYBINTEGER MYCHARACTER MYDECIMAL MYFIXED MYFLOAT MY
Se Line=2 Col=1 Alt=1,1;1 Size=9 Recl=16384 Fmt=V Files=1 Views

Figure 23. SDE Insert Records.

Deleting Records

Record, record segment and DB2 table row delete is **not** supported by Update-in-place edit technique.

Lines of data (records, record segments or DB2 table rows) may be deleted as follow:

1. Enter prefix (line) command "D(n)" on the line (and n-1 lines that follow) to be deleted.
2. Enter prefix (line) command "DD" on the first and last line of a block of lines to be deleted.
3. Execute the DELETE primary command to delete one or more consecutive, excluded and/or non-excluded lines. Unless a range of lines is specified, the first line of data to be deleted is that which occupies the focus line.

   e.g. Delete all excluded lines that fall within a range of lines denoted by label names .FTXT and .FLST.

```
DELETE  ALL  EX    .FTXT  .FLST
```

Delete treats typed and untyped lines of data identically. i.e. it is not sensitve to any assigned record-type. Following a delete operation, record-type assignment processing occurs for all remaining records affected by the operation.

Records that are SUPPRESSED or NOT SELECTED are unaffected by delete operations.

Moving Records

Moving records and record segments is **not** supported by Update-in-place edit technique. Although supported by DB2 table edit, moving table rows has no bearing on the location of a row within a DB2 table.

Lines of data (records, record segments or DB2 table rows) may be moved as follow:

1. Enter prefix (line) command "M(n)" on the line (and n-1 lines that follow) to be moved.

2. Enter prefix (line) command "MM" on the first and last line of a block of lines to be moved.

The move operation treats typed and untyped lines of data identically. i.e. it is not sensitve to any assigned record-type. Following a move operation, record-type assignment processing occurs for all records affected by the operation.

Having marked a line or block of lines to be moved, position the cursor in the prefix area of the line where the lines are to be moved and enter "A" (After) or "B" (Before). The marked line or block of lines are moved after or before the target line respectively and the cursor is placed on the first character of the first line moved.

If the target of the move operation is within the block of lines marked for move, then an error message is returned.

Only excluded or non-excluded records or segments may be moved. Records that are SUPPRESSED or NOT SELECTED are unaffected by move operations.

Copying Records

Copying records, record segments and DB2 table rows is **not** supported by Update-in-place edit technique.

Lines of data (records, record segments or DB2 table rows) may be copied as follow:

1. Enter prefix (line) command "C(n)" on the line (and n-1 lines that follow) to be copied.
2. Enter prefix (line) command "CC" on the first and last line of a block of lines to be copied.

The copy operation treats typed and untyped lines of data identically. i.e. it is not sensitve to any assigned record-type. Following a copy operation, record-type assignment processing occurs for all records affected by the operation.

Having marked a line or block of lines to be copied, position the cursor in the prefix area of the line where the lines are to be copied and enter "A" (After) or "B" (Before). The marked line or block of lines are copied after or before the target line respectively and the cursor is placed on the first character of the first line copied.

If the target of the copy operation is within the block of lines marked for copy, then an error message is returned.

Only excluded or non-excluded records or segments may be copied. Records that are SUPPRESSED or NOT SELECTED are unaffected by copy operations.

Duplicating Records

Duplicating records, record segments and DB2 table rows is **not** supported by Update-in-place edit technique.

Lines of data (records, record segments or DB2 table rows) may be duplicated as follow:

1. Enter prefix (line) command "R(n)" on the line to be duplicated (*n* number of times).
2. Enter prefix (line) command "RR(n)" on the first line "RR" on the last line of a block of lines to be duplicated (*n* number of times).
3. Enter prefix (line) command "RR(n)" on the first line "RR" on the last line of a block of lines to be duplicated (*n* number of times). Enter, the DUPLICATE primary command to duplicate the focus line a specific number of times. e.g.

```
DUPLICATE  3
```

The duplicate operation treats typed and untyped lines of data identically. i.e. it is not sensitve to any assigned record-type. Following a duplicate operation, record-type assignment processing occurs for all records affected by the operation.

Following the duplicate operation, the duplicated lines appear immediately following the last line selected for duplication and the cursor is placed on the first character of the first duplicated line.

Only excluded or non-excluded records or segments may be duplicated. Records that are SUPPRESSED or NOT SELECTED are unaffected by duplicate operations.

Undo and Redo Changes

A new change level is added for each field whose contents have changed since the last 3270 transaction.

Having made changes to field data during the edit session, the UNDO primary command (assigned to <PF22> by default) may be used to undo the change represented by the last change level. Repeatedly executing UNDO will undo each previous change in the change level stack.

Where changes have been undone, they may be re-applied, one change level at a time, using the REDO primary command (assigned to <PF23> by default.)

The number of change levels that may be undone is represented by the third number in the "Alt=n,n;m" alteration count displayed in the status line of the MDI parent window. If this number has an "*" (asterisk) appended, then this indicates that change levels are available to be re-applied with REDO.

Note: If updates are made to field data after undoing changes, then those change levels can no longer be re-applied and the trailing "*" on the alteration count is dropped.

The maximum number of change levels that can be stored for each individual file edited with SDE is set by the SET UNDOING option. This command also controls the maximum amount of storage that may be allocated for storing information required by SDE to UNDO and REDO a change represented by a change level.

Once the maximum amount of defined change levels or change storage is reached, SDE silently discards as much of the oldest change level information as is required to continue editing data. Consequently, changes represented by these discarded change levels can no longer be undone.

SDE Concepts

Although SELCOPY/i SDE runs within the CBLe text edit frame window, it is a much more sophisticated method of editing data.

Presentation and edit of record data within SDE must adhere strictly to precedents defined by the associated file structure. In order to do this, SDE introduces certain concepts and terminology that are used in its interpretation of data and are referenced throughout this documentation.

It is recommended that users familiarise themselves with the terms detailed in this section prior to performing any advanced edit operations.

Record Structure Definition

In order to display a file's data records correctly within SELCOPY/i SDE, an appropriate data definition must exist that accurately defines fields within the data records.

For SELCOPY/i SDE, a data definition is called the **structure** and must exist in an internal SDE format generated by the CREATE STRUCTURE command. Any existing COBOL or PL1 copybooks that map the file's data records may be used to generate the SDE structure.

Structure Definition Object

The Structure Definition Object (SDO) is an in-storage copy of an SDE structure comprising one or more Record Type Objects used to format data records within a file.

An SDO may be temporary or non-temporary and is referenced by its structure name. If non-temporary, the structure name is the fileid of the Structure Definition File from which the SDO is loaded and to which any changes to the SDO will be saved.

An SDO is created or loaded into storage when a structure is created (explicitly or implicitly via a CREATE STRUCTURE operation) or when a structure is nominated on an EDIT or BROWSE command.

If updates are made to Record Type Objects belonging to a non-temporary SDO, then a prompt to save the SDO will be displayed when an attempt is made to drop the SDO from storage.

Record Type Object

A Record Type Object (RTO) is a single record structure definition within an SDO which contains the following information:

1. Field offset, length and data format of all fields within a data record.
2. USE WHEN expression which defines selection criteria that identify the data records to which the RTO will apply.

So that its data can be formatted correctly, each structured record within a data set should have characteristics that match the selection criteria defined by an RTO.

Each RTO is referenced by a name which is unique within the SDO. This name is also referred to as the **record-type** and is allocated to an RTO when it is defined (CREATE STRUCTURE). The RTO record-type may be used as a generic description of those data records that are assigned that specific RTO. (See *"Record Type Assignment"* .) e.g. Client1 data records are considered to be records that are assigned an RTO with the name "Client1".

An RTO (and therfore the SDO) may be altered (e.g. via the USE WHEN command) and saved to a Structure Definition File.

Record Type Object Data Length

The RTO data length is the sum of all field lengths defined by the RTO.

If all RTO fields are of fixed length, the RTO data length is a fixed value.

If, however, the RTO includes variable length field definitions or a vriable number of repeating field groups, the RTO data length will encompass a range of possible values. In this case, the RTO data length is variable and has a minimum and maximum value.

The RTO data length is used in record-type assignment processing.

Structure Definition File

A Structure Definition File (SDF) is a disk file (sequential data set or PDS/PDSE library member) containing a saved (i.e. non-temporary) copy of a Structure Definition Object.

The SDF fileid is also the structure name assigned to the non-temporary SDO.
On executing an EDIT or BROWSE command, if the specified SDO is not already loaded in storage, it gets loaded from the SDF.

If a non-temporary SDO is altered during an SDE edit session, then the user will be prompted to save the SDO to its SDF when the SDO is dropped from storage. Note that a save is performed automatically on explicit execution of the CREATE STRUCTURE command for a non-temporary SDO.

Record Type Assignment

Where a structure has been specified for SDE processing of data records (e.g. SDE EDIT, BROWSE, File Compare, etc.) then, on initial load of the data records or following execution of a USE WHEN/ALWAYS/NEVER command, record-type assignment processing is performed.

The record-type (RTO) assigned to an individual record is determined using the following order of precedence. Note that, unless specified by a USE ALWAYS ON condition, an RTO for which the **USE NEVER** condition has been set **ON** will be excluded from any RTO assignment test.

1. Use the RTO identified by a USE *record_type* ALWAYS ON condition. If the USE NEVER condition is ON for this RTO, it will be set OFF.
2. Use the first RTO in the structure definition (SDO) for which a USE WHEN expression tests "true".
3. Use the first RTO in the SDO defined as being DEFAULT.
4. Use the first RTO defined in the SDO for which the unexpanded record length matches the RTO data length. For an RTO of variable data length, the unexpanded record length is considered a match if it falls within the minimum and maximum RTO data length values.
   Note that, having found a match, no other RTOs in the SDO are tested.
5. Use the first RTO in the SDO that is of a **fixed** RTO data length.
6. Use the first RTO in the SDO.
7. Use the default record type "UnMapped" which consists of a single field "UnMapped" of maximum length equal to the data set's LRECL value, and with a data type AN (alpha-numeric.)

Specification of USE record-type criteria allows assignment of that RTO to a record having a length that does not necessarily match the RTO data length. If this is the case, only areas of the record that are formatted by the assigned RTO will be displayed.

Any unformatted record data that follows the formatted data of the same record is not displayed but is preserved if changes are made to the formatted data. The unformatted data may be displayed and changed at any time using a different display format. e.g. FORMAT CHAR or FORMAT HEX.

Where unformatted data exists at the end of a record, the flag "=LGTH>" is displayed in the prefix area.

Structured Data Fields

Records that have been assigned a record type (RTO) are displayed with the record data presented in individual formatted field columns.

The names assigned to each field are those defined by the SDO, COBOL copybook or PL1 include file. Similarly, the assigned field reference number reflects the order in which the field occurs in the record structure hierarchy. Both are displayed in the record type headers (see *SDE Window Views*.)

SDE commands and expressions may reference a field via its field name or its field reference number.

Field Name

If unique, the field may be identified simply by specifying its defined field name.

However, if the field name is non-unique, occuring in one or more sub-structures (COBOL data description groups) that exist in the record type definition, then the field name must be fully or partially qualified by the the structure names of each level of structure in the hierarchy to which the field belongs. The default qualifier separator symbol is '.' (dot/period) but this may be tailored using the SET QSEPARATOR option. e.g. "structA.structN.fieldX".

The number of preceding structure name qualifiers specified before the field name need only be enough to uniquely identify the required field. e.g. "structN.fieldX" is enough to destinguish it from "structM.fieldX".

If desired, the field name may contain qualifiers of every level in the hierarchy including the record type name, which is itself a structure definition. e.g. "OrderRec.structA.structN.fieldX".

Note that, unless parameter CASE(RESPECT) was used to create the structure (SDO), the field name and any of its qualifier structure names, may be typed in any character case.

Field Reference
The field reference number is unique in the record type definition.

The required field may be identified by specifying its field reference number prefixed by a "#" (hash) symbol.

Field Subscripts
Fields that constitute elements of an array (COBOL OCCURS clause) may be individually identified using a parenthesised, comma separated list of array vectors as a subscript to the array field name or field reference.

One array vector value must be specified for each dimension defined to the array. Each vector is an integer value identifying the field's position in that dimension of the array. e.g. #13(3) is element 3 of a single dimension field array, RoomArea(6,2) is the x,y coordinate of an element in a two dimensional field array.

SDE Window Views

An SDE window is an MDI child window (view) that may be opened from within the CBLe Text Editor or SELCOPY Interactive application parent window.

Data within an SDE window is presented as either a multi record or single record view.

Multi Record View

This is an SDE MDI child window containing multiple records displayed horizontally so that each record occupies a single line of the display.

Field Column-Heading Lines

By default, a group of 5 field column-heading lines are displayed within the window display area immediately above the first data record in view. Unless records associated with a different record type are visible (see Record Data Display below), no further groups of column-heading lines are displayed within the current display area view.

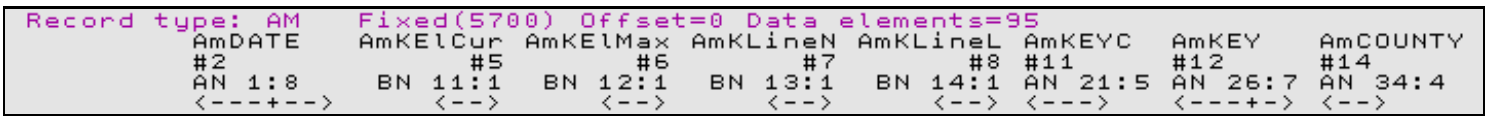

Record type: AM Fixed(5700) Offset=0 Data elements=95
AmDATE AmKElCur AmKElMax AmKLineN AmKLineL AmKEYC AmKEY AmCOUNTY
#2 #5 #6 #7 #8 #11 #12 #14
AN 1:8 BN 11:1 BN 12:1 BN 13:1 BN 14:1 AN 21:5 AN 26:7 AN 34:4

Figure 24. SDE Multi-Record Display Header Lines.

Header Line 1: Record Type
For SDE display of file records, this header identifies:

1. The record-type assigned to the records (or record segments) that follow.
2. The record type format (Fixed or Variable) followed by the record type length (i.e. the sum of the record type field lengths) in "( )" (parentheses). Note that, for variable record types a maximum,minimum length pair is displayed.
3. The offset into the record at which record type begins to format the data. This may be a positive or negative value. (Offset=*n*)
4. The number of data elements (field references) within the record type definition. This number includes the record type STRUCTURE field, and any other STRUCTURE fields, and so will be greater than the number of fields in view.

For SDE DB2 table row display, this header identifies:

1. The record type (**Row name**) assigned to the table rows - default is the table owner ID name.
2. The record type format (Fixed or Variable) followed by the record type length (i.e. the sum of the record type field lengths) in "( )" (parentheses). Note that, for variable record types a maximum,minimum length pair is displayed.
3. The number of columns defined by the DB2 results table.

Header Line 2: Field Name
Where present, the names of fields selected for display, are used as the column header names and are located in the second line of the column header.
This header line may not be switched off.

Header Line 3: Field Reference

The third column header line displays the field number (reference) within the record. By default, when table format is specified, the fields displayed will be the lowest level of a nested group of fields. Therefore, the names of fields that define a group of fields, will not be displayed and so their corresponding field numbers will also be missing from the sequence of displayed fields.
This header line may be switched on/off using the SET REFERENCE command.

Header Line 4: Field Type

For DB2 table edit, the fourth column header line displays the field data type as specified in the table column definition.

For data files, the fourth column header line displays the field structure (type) as a 2-character data type code followed by the field's start position (byte number within the record) and the field length. For fields whose lengths are defined in number of bits, the field start position includes an offset into the byte following a "." (decimal point) and the unit for field length is bits. The start position and length displays are separated by a ":" (colon).

Recognised file editing data type codes are as follow:

AN	CHARACTER, VARCHAR, XVARCHAR, STRUCTURE or UNION
BI	BINTEGER
BN	INTEGER
BT	BIT
FB	FIXEDHEX, FIXEDBIN
FP	FLOAT
PD	DECIMAL
X	HEXADECIMAL
ZD	ZONED

See *"SDE Data Types"* for descriptions of each of the supported data types and their representation in the Field Type header line.
This header line may be switched on and off using the SET TYPE command.

Header Line 5: Scale

The fifth column header line displays a scale for each field in the display. The width of the scale is equivalent to the width of the field display area which is wide enough to display all the field data or, for numeric fields, the largest possible numeric value.

Where the display field is wider than 2 bytes, a scale begins with "<" (less than) as position 1 and ends with ">" (greater than) as the last position of the field.
The scale characters displayed and the the ability to switch this header line on and off is controlled by the SET SCALE command.

Prefix Area

By default, the record prefix is set on and so the prefix area is displayed containing the record sequence numbers. The record prefix area may tailored or set on/off using the SET PREFIX command.

Data Records

By default, all records of the file, regardless of assigned record type, are selected for display.
Records that have been selected for display are identified as **data records**.

Using the VIEW command, records of different record types may be displayed or suppressed within a multi record view.

Where multiple record types are visible, each record in the window display area that is of a different record type to the previous record, will be immediately preceded by its column headers. Therefore, when viewing a mixture of records of different record types, the display area may contain multiple groups of column header lines. It is also possible that only part of a group of column header lines is displayed at the bottom of the display area.

Within formatted data records, the display of character (AN, CH or VARCHAR) fields that contain non-printable characters differs from the rest of the displayed fields in both multi record and single record views. Printable characters in these fields are displayed in a different colour with underscore highlighting so that non-printable characters appear as immoveable gaps, effectively fixing the length of printable character data before them. Only printable characters in these fields may be overtyped. Non-printable characters may only be updated using the CHANGE command or by setting HEX ON, so allowing overtype of the hex display of the data.

Scrolling left and right will scroll the fields within all records of the default record type only.
Scrolling up and down will scroll through all records of the file.

Shadow Lines

A record or group of consecutive records that are not displayed because they are suppressed, specifically excluded by the user or have no associated RTO (not selected) are, by default, replaced by a **shadow line**.

A shadow line specifies the number of consecutive records excluded from the display, the reason for their exclusion ("suppressed", "excluded" or "not selected") and, if applicable, the associated record type.
Shadow lines may be set on/off using the SET SHADOW command.

Single Record View

An SDE MDI child window containing data from a single record displayed vertically so that each field of the record occupies a single line of the display.

A group of 3 field column-heading lines are displayed within the window display area immediately above the first field in view. Header line 1 contains the record sequence number; Header line 2 the record type (as illustrated in Header line 1 of multi record view) and Header line 3 contains the column headings **Field** and **Data**.

The **Field** column displays the field name (if defined), the level of nesting of the field within field groups and the field structure (type) as illustrated above in header line 4 of the multi record view.
The **Data** column displays the data within the record for the specific field.

Each line of data identifies an individual field and its equivalent value, collectively referenced as a **field data line**.

Records that are suppressed, excluded or not selected cannot be viewed within a single record view.

Scrolling left and right will display the previous and next records respectively that have not been suppressed.
Scrolling up and down will display the previous and next fields within the record respectively.

SDE Data Formats

Data within an SDE window view is displayed in one of the following data formats.

Table Format

Forces a multi record window view. If table format is selected on EDIT or BROWSE, then records are assigned an appropriate record type (RTO) and formatted so that record data is split into its component fields as defined by the RTO. The data is processed so that all fields are displayed as printable character, numeric data fields having first been converted to decimal.

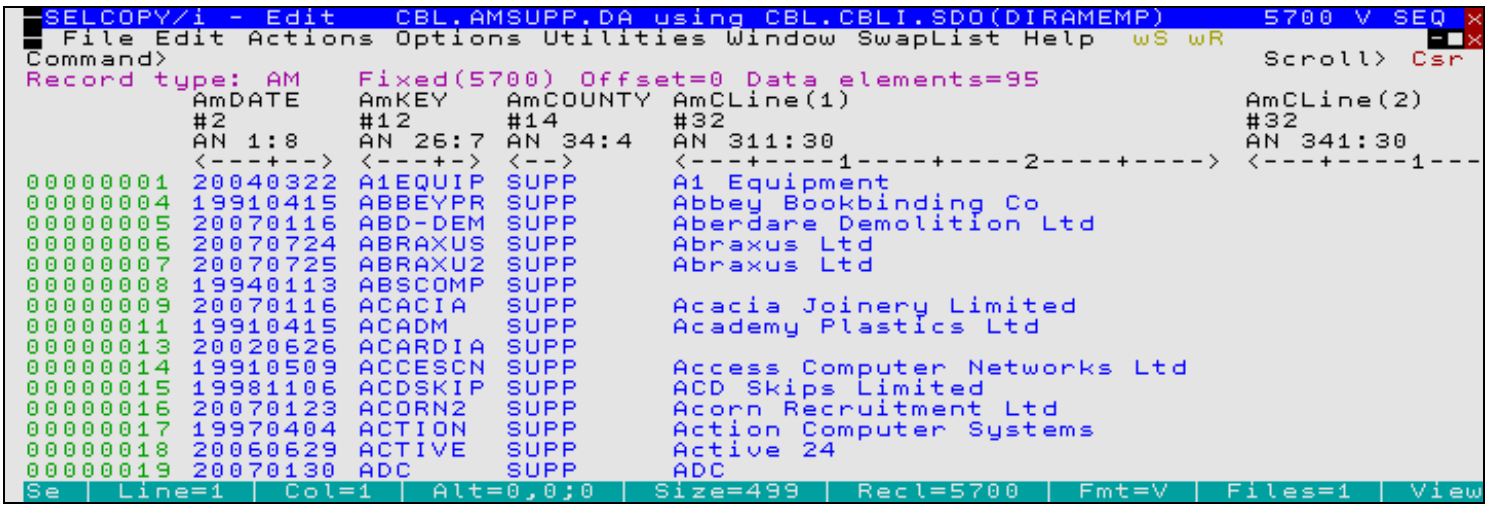

SELCOPY/i - Edit CBL.AMSUPP.DA using CBL.CBLI.SDO(DIRAMEMP) 5700 V SEQ
File Edit Actions Options Utilities Window SwapList Help wS wR
Command>
Scroll> Csr
Record type: AM Fixed(5700) Offset=0 Data elements=95
AmDATE AmKEY AmCOUNTY AmCLine(1) AmCLine(2)
#2 #12 #14 #32 #32
AN 1:8 AN 26:7 AN 34:4 AN 311:30 AN 341:30
00000001 20040322 A1EQUIP SUPP A1 Equipment
00000004 19910415 ABBEYPR SUPP Abbey Bookbinding Co
00000005 20070116 ABD-DEM SUPP Aberdare Demolition Ltd
00000006 20070724 ABRAXUS SUPP Abraxus Ltd
00000007 20070725 ABRAXU2 SUPP Abraxus Ltd
00000008 19940113 ABSCOMP SUPP
00000009 20070116 ACACIA SUPP Acacia Joinery Limited
00000011 19910415 ACADM SUPP Academy Plastics Ltd
00000013 20020626 ACARDIA SUPP
00000014 19910509 ACCESCN SUPP Access Computer Networks Ltd
00000015 19981106 ACDSKIP SUPP ACD Skips Limited
00000016 20070123 ACORN2 SUPP Acorn Recruitment Ltd
00000017 19970404 ACTION SUPP Action Computer Systems
00000018 20060629 ACTIVE SUPP Active 24
00000019 20070130 ADC SUPP ADC
Se Line=1 Col=1 Alt=0,0;0 Size=499 Recl=5700 Fmt=V Files=1 View

Figure 25. SDE Table Format.

Single Format

Record data is processed as for a table view except that the display becomes a single record window view of the default record.

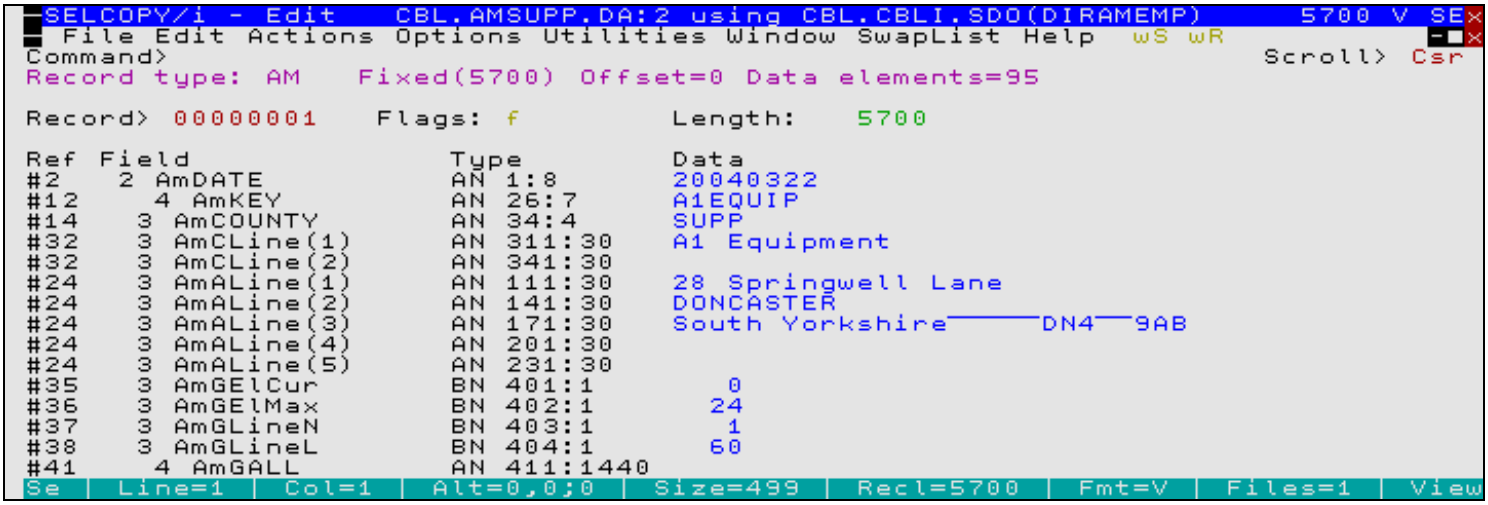

SELCOPY/i - Edit CBL.AMSUPP.DA:2 using CBL.CBLI.SDO(DIRAMEMP) 5700 V SE
File Edit Actions Options Utilities Window SwapList Help wS wR
Command>
Scroll> Csr
Record type: AM Fixed(5700) Offset=0 Data elements=95
Record> 00000001 Flags: f Length: 5700
Ref Field Type Data
#2 2 AmDATE AN 1:8 20040322
#12 4 AmKEY AN 26:7 A1EQUIP
#14 3 AmCOUNTY AN 34:4 SUPP
#32 3 AmCLine(1) AN 311:30 A1 Equipment
#32 3 AmCLine(2) AN 341:30
#24 3 AmALine(1) AN 111:30 28 Springwell Lane
#24 3 AmALine(2) AN 141:30 DONCASTER
#24 3 AmALine(3) AN 171:30 South Yorkshire DN4 9AB
#24 3 AmALine(4) AN 201:30
#24 3 AmALine(5) AN 231:30
#35 3 AmGElCur BN 401:1 0
#36 3 AmGElMax BN 402:1 24
#37 3 AmGLineN BN 403:1 1
#38 3 AmGLineL BN 404:1 60
#41 4 AmGALL AN 411:1440
Se Line=1 Col=1 Alt=0,0;0 Size=499 Recl=5700 Fmt=V Files=1 View

Figure 26. SDE Single Format.

Character Format

Record data in the current multi or single record view is mapped as a single, variable length character field with field name "UnMapped" or, for record segments, "UnMappedSeg". No conversion of numeric or unprintable data is performed and the window view (either multi or single record) remains unchanged.

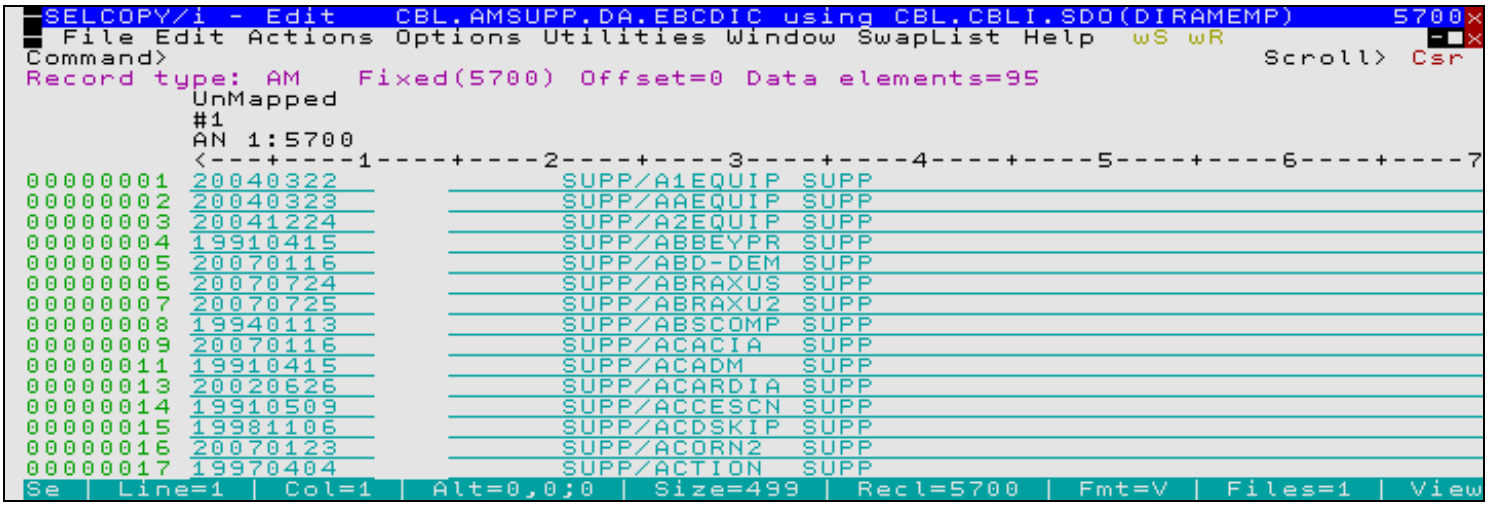

SELCOPY/i - Edit CBL.AMSUPP.DA.EBCDIC using CBL.CBLI.SDO(DIRAMEMP) 5700
File Edit Actions Options Utilities Window SwapList Help wS wR
Command>
Scroll> Csr
Record type: AM Fixed(5700) Offset=0 Data elements=95
UnMapped
#1
AN 1:5700
00000001 20040322 SUPP/A1EQUIP SUPP
00000002 20040323 SUPP/AAEQUIP SUPP
00000003 20041224 SUPP/A2EQUIP SUPP
00000004 19910415 SUPP/ABBEYPR SUPP
00000005 20070116 SUPP/ABD-DEM SUPP
00000006 20070724 SUPP/ABRAXUS SUPP
00000007 20070725 SUPP/ABRAXU2 SUPP
00000008 19940113 SUPP/ABSCOMP SUPP
00000009 20070116 SUPP/ACACIA SUPP
00000011 19910415 SUPP/ACADM SUPP
00000013 20020626 SUPP/ACARDIA SUPP
00000014 19910509 SUPP/ACCESCN SUPP
00000015 19981106 SUPP/ACDSKIP SUPP
00000016 20070123 SUPP/ACORN2 SUPP
00000017 19970404 SUPP/ACTION SUPP
Se Line=1 Col=1 Alt=0,0;0 Size=499 Recl=5700 Fmt=V Files=1 View

Figure 27. SDE Character Format.

Hexadecimal Format

Record data in the current multi or single record view is displayed as for character format with the addition that the hexadecimal representation of the data is displayed below the character data.

Note that the Hex representation occupies an additional 2 lines of the display for each record and consists of characters 0-9, A-F (indicating 4-bit binary values, B'0000'-B'1111'). The first line contains the high order 4 bits of the byte and the second line contains the low order 4 bits.

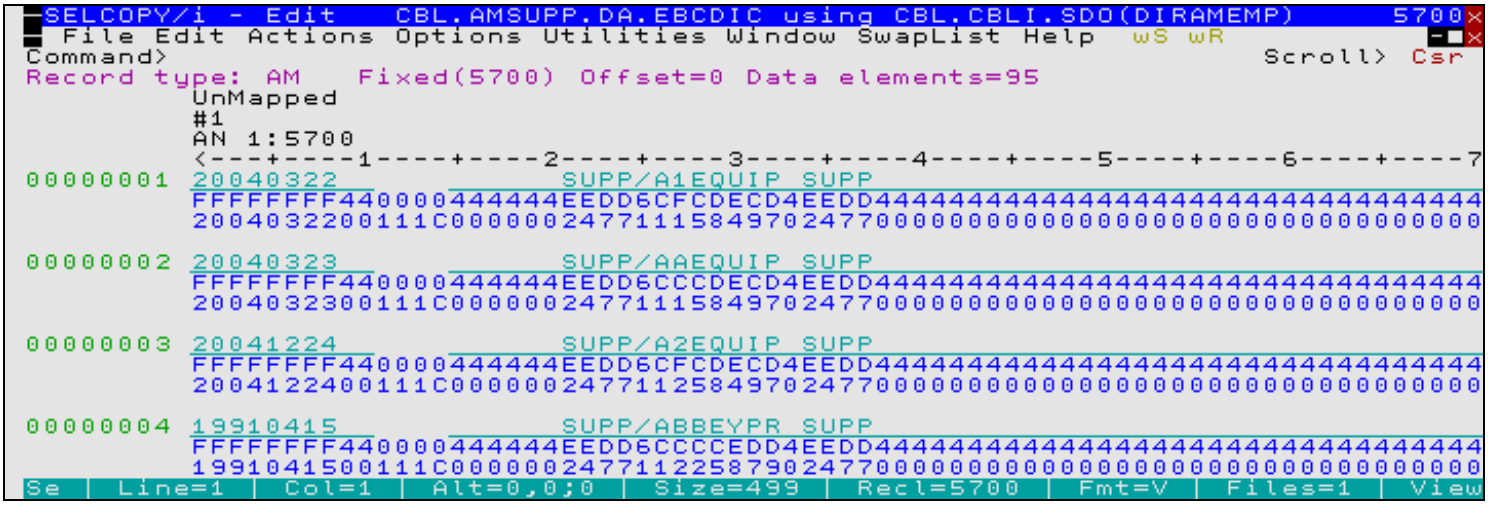

SELCOPY/i - Edit CBL.AMSUPP.DA.EBCDIC using CBL.CBLI.SDO(DIRAMEMP) 5700
File Edit Actions Options Utilities Window SwapList Help wS wR
Command>
Scroll> Csr
Record type: AM Fixed(5700) Offset=0 Data elements=95
UnMapped
#1
AN 1:5700
<---+----1----+----2----+----3----+----4----+----5----+----6----+----7
00000001 20040322 SUPP/A1EQUIP SUPP
00000002 20040323 SUPP/AAEQUIP SUPP
00000003 20041224 SUPP/A2EQUIP SUPP
00000004 19910415 SUPP/ABBEYPR SUPP
Se | Line=1 | Col=1 | Alt=0,0;0 | Size=499 | Recl=5700 | Fmt=V | Files=1 | View

Figure 28. SDE Hexadecimal Format.

Hexadecimal Dump Format

Record data is displayed without any record-type formatting, in a single record window hex dump view of the default record.

On the left of the display is a field containing the position of the data within the record (in hex or decimal representation.)

Following this field, the record data is represented by a number of blank delimited fullwords (groups of 4 bytes) in long hex format, followed by the equivalent number of bytes in character format.

Long hex format is where the hexadecimal representation of a character occupies two characters of the display. e.g. C'A' is represented as C'C1' in long hex format.

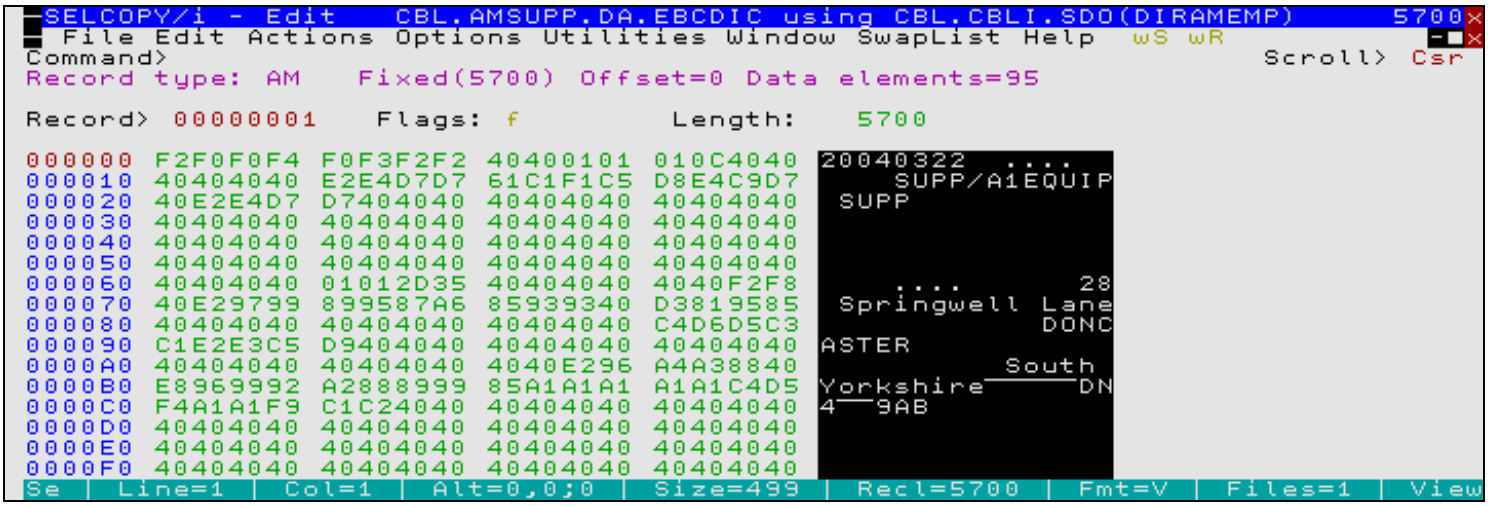


Figure 29. SDE Hexadecimal Dump Format.

SDE Data Types

Structured records may be mapped by a record type object (RTO) whereby fields may be interpreted as being of any of the the following supported field types:

Bit Flags

A Binary field of length specified in number of bits where each bit is interpreted as a flag switch which is either ON (1) or OFF (0). Fields of this data type have a data type record header of **"BT *ppp*:*nn*"** where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bits).

Binary Data Fixed

A **DB2** Binary data field (built-in data type BINARY) of fixed length whose contents are interpreted as printable ASCII or EBCDIC character data. Fields of this data type have a data type record header of **"BINARY(*nn*)**" where *nn* is the length of the field (in bytes).

Binary Data Variable

A **DB2** Binary data field (built-in data type VARBINARY) of variable length contents are interpreted as printable ASCII or EBCDIC character data. Fields of this data type have a data type record header of **"VARBIN(*nn*)**" where *nn* is the maximum length of the field (in bytes).

Character Fixed

A Character field of fixed length specified in number of bytes whose contents are interpreted as printable ASCII or EBCDIC character data. Fields of this data type have a data type record header of **"AN *ppp*:*nn***" or, for **DB2** table edit/browse, "**CH(*nn*)**" where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bytes).

Character Variable (Field of Varying Length with Nominated Length Field)

A Character field of variable length specified in number of bytes by a nominated field within the record data. The nominated field may be of any numerical data type and the character field contents are interpreted as printable ASCII or EBCDIC character data. Fields of this data type have a data type record header of **"AN *ppp*:*nn***" where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bytes).

Character Variable (Field of Varying Length with Preceding Length Field)

A Character field of variable length specified in number of bytes by an Integer Binary (Byte) field located immediately before the character data. The Integer Binary field is of a predefined length which may be defined as being included with or excluded from the character field length. The character field contents are interpreted as printable ASCII or EBCDIC character data. Fields of this data type have a data type record header of **"AN *ppp*:*nn***" or, for **DB2** table edit/browse, "**VARCHAR(*nn*)**" where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bytes).

Character Variable (Field of Static Length with Preceding Length Field)

A Character field of variable length padded with blanks to occupy a field of a fixed, predefined length. The variable length of the character data is determined by a 2-byte Integer Binary (Byte) field located immediately before the character data. The 2-byte Integer Binary field length is not included within the stored length. This data type is equivalent to PL/1 fields declared as CHARACTER VARYING. The character field contents are interpreted as printable ASCII or EBCDIC character data. Fields of this data type have a data type record header of **"AN *ppp*:*nn***" where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bytes). This value includes the 2-byte length field.

Character Variable (Field of Static Length with Null Termination)

A Character field of variable length occupying a field of a fixed, predefined length. The end of the variable length character data is determined by at least one null (x'00') termination character. Therefore, in order to support a character strings of the maximum field length, the predefined fixed field length is always 1 byte longer than the maximum length requested. This data type is equivalent to PL/1 fields declared as CHARACTER VARYINGZ. The character field contents are interpreted as printable ASCII or EBCDIC character data. Fields of this data type have a data type record header of **"AN *ppp*:*nn***" where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bytes). This value includes the additional 1-byte.

Date

A Date field displayed in ISO format with separator character "/" (yyyy/mm/dd) or, for **DB2** table edit/browse, in the locally defined DB2 date string format. The internal format of the date string for non-DB2 edit/browse may be that returned by the TIME Assembler macro for packed or binary data, or the internal date format of ICF catalog records or VTOC format 1/3 records. Fields of this data type have a data type record header of **"DT**" or, for DB2 table edit/browse, "**DATE**".

Fixed Point Binary

A Binary field defined as having a precision and scale specified in number of decimal digits, and whose value is displayed as a fixed point decimal number. The length of the binary field is either 2-bytes (Halfword), 4-bytes (Word) or 8-bytes (Doubleword) as implied by the precision which defines the total number of decimal digits. i.e. precision <5 implies halfword, precision >=5 and <10 implies word, precision >=10 implies doubleword. The scale defines the number of digits to the right of the decimal point. Fields of this data type have a data type record header of **"FB *ppp*:*nn***" where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bytes).

Fixed Point Packed Decimal

A Packed Decimal field defined as having a precision and scale specified in number of decimal digits, and whose value is displayed as a fixed point decimal number. The length of the packed decimal field is implied by the precision which defines the total number of decimal digits. The scale defines the number of digits to the right of the decimal point. Fields of this data type have a data type record header of **"PD *ppp*:*nn***" or, for **DB2** table edit/browse, "**DEC(*pr*,*sc*)**" where *ppp* is the position of the field within the expanded record (in bytes), *nn* is the length of the field (in bytes), *pr* is the defined precision and *sc* the defined scale.

Fixed Point Zoned Decimal

A Zoned Decimal Character field defined as having a precision and scale specified in number of bytes, and whose value is displayed as a fixed point decimal number. The length of the zoned decimal field is implied by the precision which defines the total number of decimal digits. The scale defines the number of digits to the right of the decimal point. Fields of this data type have a data type record header of **"ZD *ppp*:*nn***" where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bytes).

Floating Point Hexadecimal

A Hexadecimal Floating Point field of length 4 bytes (short) or 8 bytes (long) whose value is displayed as a decimal number consisting of a normalised mantissa and exponent. Fields of this data type have a data type record header of **"FP *ppp*:*nn***" or, for **DB2** table edit/browse, "**REAL**" for short fields and "FLOAT" for long fields where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bytes).

Floating Point Binary

A Binary Floating Point (IEEE 754) field of length 4 bytes (short) or 8 bytes (long) whose value is displayed as a decimal number consisting of a normalised mantissa and exponent. Fields of this data type have a data type record header of **"FP *ppp*:*nn***" where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bytes).

Floating Point Decimal

A Decimal Floating Point (IEEE 754r) field of length 8 bytes or 16 bytes whose value is displayed as a decimal number consisting of a normalised mantissa and exponent. Fields of this data type have a data type record header of **"FP *ppp*:*nn***" or, for **DB2** table edit/browse, "**DECFLOAT(*pr*)**" where *ppp* is the position of the field within the expanded record (in bytes), *nn* is the length of the field (in bytes) and *pr* is the defined precision (16 or 34).

Hexadecimal

A Hexadecimal field of a fixed length specified in number of bytes whose contents are displayed as a printable hexadecimal character string. Fields of this data type have a data type record header of **"X *ppp*:*nn***" where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bytes).

Integer Binary (Bit)

A Binary field of length specified in number of bits, the contents of which are interpreted as being numeric and whose value is displayed as an integer decimal number. Fields of this data type have a data type record header of **"BI *ppp*:*nn***" where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bits). Compare with Integer Binary (Byte) data type which is a binary integer field with length expressed in number of bytes.

Integer Binary (Byte)

A Binary field of length specified in number of bytes (maximum 8-bytes), the contents of which are interpreted as being numeric and whose value is displayed as an integer decimal number. Fields of this data type have a data type record header of **"BN *ppp*:*nn***" or, for **DB2** table edit/browse, "**SMINT**" (halfword), "**INT**" (word) or "**BIGINT**" (doubleword) where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bytes). Compare with Binary Integer (Bit) data type which is a binary integer field with length expressed in number of bits.

PL/1 Picture String (Character)

A PL/1 style PICTURE string representing a character data item (i.e. no numerical interpretation) of length determined by the specified picture string. The picture string may contain any valid PL/1 picture character for character data. Fields of this data type have a data type record header of **"AN *ppp*:*nn***" where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bytes).

PL/1 Picture String (Fixed Point Numerical)

A PL/1 style PICTURE string representing a FIXED numerical character data item of length determined by the specified picture string. The picture string may contain any valid PL/1 picture character for numeric character data except for exponent characters "E" and "K". Fields of this data type have a data type record header of **"AN *ppp*:*nn***" where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bytes).

PL/1 Picture String (Floating Point Numerical)

Not Yet Supported
A PL/1 style PICTURE string representing a FLOAT numerical character data item of length determined by the specified picture string. The picture string may contain any valid PL/1 picture character for numeric character. Fields of this data type have a data type record header of **"AN *ppp*:*nn***" where *ppp* is the position of the field within the expanded record (in bytes) and *nn* is the length of the field (in bytes).

Stucture

A Structure consisting of one or more fields each defined as being any of the supported data types. The structure field length is defined as being the sum of all field lengths within the structure. A field within a structure may itself be of the structure data type, so defining a structure nested within a structure.

Time

A Time field displayed with separator character ":" (HH:MM:SS) or, for **DB2** table edit/browse, in the locally defined DB2 time string format. The internal format of the time string for non-DB2 edit/browse may be that returned by the TIME Assembler macro for packed or binary data. Fields of this data type have a data type record header of **"TI**" or, for DB2 table edit/browse, "**TIME**".

Timestamp

A Timestamp (Date and Time) field displayed in ISO format with date separator character "/" (yyyy/mm/dd) and time separator character ":" (HH:MM:SS), or, for **DB2** table edit/browse, in the locally defined DB2 timestamp string format. The internal format of the date and time string for non-DB2 edit/browse may be that returned by the TIME Assembler macro for packed or binary data, the system TOD clock (returned by the STCK operation) or an HFS file directory entry. Fields of this data type have a data type record header of **"TS**" or, for DB2 table edit/browse, "**TIMESTAMP**".

Union

A Union of one or more fields of any of the supported data types. Each field in the union occupies the same start position in the data record. The union field length is defined as being the length of the longest field in the union. Unions allow the same data to be interpreted as more than one data-type.

SDE Edit Techniques

SDE data edit employs sophisticated techniques for in-storage record data management. These techniques allow for more control over the data being edited and also allows the user to edit data sets that would not otherwise fit in storage.

Prior to edit, SELCOPY/i SDE establishes the user supplied edit options, the data set organisation and size, and the amount of storage available. Based on these, SDE will perform the appropriate editing technique as follows:

Full Edit

Supported for sequential data sets, PDS(E) members, DB2 tables and VSAM RRDS and ESDS files defined with option REUSE, all records belonging to the data set are loaded into storage prior to display in an SDE window view. Full edit supports change, insert, delete, copy and move of data records. Furthermore, records may be updated to alter the record length if supported by the data set organisation or structure (SDO).

This edit technique will be used where conditions for Update-in-place and KSDS edit are not satisfied. Before a data set is opened for Full Edit, SDE first establishes whether enough storage is available to load the entire data set. For non-DB2 table edit, if insufficient storage is available or AUXILIARY is specified on the EDIT command, AUXILIARY edit will be used instead. For DB2 table edit, this will result in a storage error.

Although Full Edit can result in a delay as the data set is loaded into storage, subsequent edit operations that involve scrolling or data scanning will operate faster. When the data set is saved, the entire data set is re-written back to disk using the in-storage copy of the records.

If Full Edit is in effect, "Edit" is displayed in the SDE window title bar to the left of the DSN.

Read Only Edit

Supported for sequential data sets, PDS(E) members, DB2 tables and VSAM RRDS and ESDS files, all records belonging to the data set are loaded into storage prior to display in an SDE window view.

Read Only edit supports all functionality provided by Full Edit except that changes cannot be saved to disk unless the fileid is first updated (using SET DSN, FILEID, FMODE, FNAME, FPATH or FTYPE) to be a non-existing fileid. Having done this, saving the data will open a dialog window to allocate or IDCAMS DEFINE the data set. This allows the user to effectively edit and create sample data before copying it to a new data set, whilst being secure in the knowledge that the orginal data set will remain intact.

Read Only edit is invoked by specifying READONLY on the SDE EDIT CLI command. Like Full Edit, AUXILIARY edit will be used if insufficient storage is available or AUXILIARY is specified on the EDIT command.

If Read Only Edit is in effect, "Edit(RO)" is displayed in the SDE window title bar to the left of the DSN. If AUXILIARY edit is used for read only editing, "Edit(AUXRO)" is displayed instead.

Update-in-place Edit

Not supported for DB2 table edit, this edit technique allows in-place record update only. Records cannot be inserted, deleted, moved or copied and record lengths cannot be altered. The exception to this is RRDS update-in-place edit where empty slots may be created by record delete and made enterable by record insert.

Update-in-place edit is invoked or implied by the following:

1. Specification of parameter UPDATE on the SDE EDIT CLI command.
2. Parameter FROM and/or FOR (indicating start record and number of records) is specified on the SDE EDIT CLI command.
3. Data set is a VSAM ESDS or RRDS file with IDCAMS DEFINE CLUSTER option NOREUSE.
4. Data set is not VSAM KSDS and load of the data set into storage has been interrupted by the user following receipt of the Load Threshold break-in modal window.

Before a data set is opened for Update-in-place Edit, SDE first establishes whether enough storage is available to load the entire data set. If insufficient storage is available, SDE will only load a single page of records for display in the SDE window view. Subsequently, only records that have been altered and records displayed in the current SDE window view are held in storage.

When the data set is saved, any record that has alterations is updated in place (replaced) by the modified, in-storage copy of the record.

If Update-in-place Edit is in effect, "Edit(UP)" is displayed in the SDE window title bar to the left of the DSN.

KSDS Edit

KSDS Edit is used for Read Only, Full Edit or Update-in-place Edit of VSAM KSDS data sets.

For KSDS Edit, only records that have been altered, inserted or deleted and records displayed in the current SDE window view are held in storage. It is possible to do this, even for Full Edit, as KSDS records may be deleted and inserted out of sequence.

The capabilities and restrictions of Read ONly Edit, Full Edit and Update-in-place Edit still apply and "Edit(RO)", "Edit" or "Edit(UP)" is displayed in the title bar as appropriate. Note that Read Only Edit or Update-in-place Edit of KSDS data sets is invoked using the READONLY or UPDATE parameter of the SDE EDIT command respectively, otherwise Full Edit is default.

AUXILIARY Edit

Auxiliary edit first makes a copy of the edit data set to a second, auxiliary file on disk.

Only records that have been altered, inserted or deleted and records in the current SDE window view are held in storage. For Auxiliary Edit, a data set can only be saved when it is closed during exit of the SDE window view. This is becuase the data set is completely re-written using a combination of records obtained from the auxiliary data set and records held in storage. Note that SAVE is not supported using this edit technique.

Auxiliary edit is used to provide full edit capabilities for sequential data sets, PDS(E) members and VSAM RRDS and ESDS files defined with option REUSE, when one of the following is true:

1. The data set is too large to be loaded into available storage.
2. Parameter AUXILIARY is specified on the SDE EDIT CLI command.

If Auxiliary Edit is in effect, "Edit(AUX)" is displayed in the SDE window title bar to the left of the DSN.

The auxiliary data set is allocated by SDE and subsequently deleted after the edited file is closed. The generated data set name has HLQ prefix as defined by the SDE.AUXDSNPREFIX INI variable (default &USER.CBLI.SDEAUX) followed by qualifiers of the format Dyyyyjjj.Thhmmsst. Note that the HLQ prefix may be updated using the SDE SET AUXDSNPREFIX command.

Changing Record Lengths

The SDE is a structured data editor and so, in order to preserve the structure of record data, the lengths of updated records are unchanged when the data is saved. This occurs whether or not a structure (SDO) or copybook has been applied to the record data and has the effect of preserving any trailing blanks that exist in the records.

However, the length of a variable length record may be altered in the following circumstances:

1. Execution of the CHANGE command results in a change to the record length. This may be a change to a numerical field value that represents the current length of a variable length field, or a change to the length of character data in a variable length field.

   In particular, use of CHANGE with parameter DATA and different length search and replace strings will result in a field length change.
   Note that, for variable length records, record type "UnMapped" is used for unformatted data and comprises a single field (field name "UnMapped") which is of variable length.

2. When the record information Length field is displayed (RECLENGTH ON or SET RECINFO ON LENGTH), then the length of a record may be altered by simply overtyping the Length value with the new record length.

   Altering the record length in this way will force a re-assessment of the record type assigned to the record data. This is done using standard record type (RTO) assignment rules.

3. When a structure (SDO) or copybook has been applied to the record data and the assigned record type contains length or array vector fields, then overtyping the values in any of these fields will also update the record length accordingly.

Attempting to alter the record length will fail if the new length is greater than the maximum LRECL defined for the file.

SDE Current Window

The current SDE window is the SDE window view on which focus was last placed. This may be the focus window or, if the focus window is not an SDE window view, the last SDE window view visited.

The concept of a current SDE window view is important when executing SDE commands from a CBLe text edit view via the SDATA command which directs the SDE command to the current SDE window view.

Using SDATA is particularly useful for issuing stored SDE commands from within a CMX file.

Default Record Type

SDE windows are capable of displaying records of more than one record type simultaneously within the same display area.

Because many SDE commands and functions operate on records of a single, specific record type, the concept of a default record type exists for use when a record type has not been specifically identified via a command parameter.

The following identifies the order in which the default record type is determined by SELCOPY/i SDE:

1. The record type of the focus line if the focus line is a visible record or an EXCLUDED shadow line.
2. The prevailing setting of DRECTYPE.

The value of DRECTYPE is assigned when the SDE window is opened and is established in the following order:

1. The record type that is the first, non-generic argument specified on the VIEW parameter of the EDIT or BROWSE command.
2. The record type of the RTO in the SDO defined as being the DEFAULT.
3. The record type of the first visible record of the edited or browsed data.
4. The record type of the first RTO in the SDO.

The DRECTYPE value may be updated by any of the following:

- SET DRECTYPE explicitly sets the DRECTYPE value.
- Execution of the VIEW command will update DRECTYPE to be the first, non-generic record type argument.

- Execution of any command that supports a *record_type* argument will update DRECTYPE to be the record type selected for the command. This is true whether the record type has been specified explicitly in the command syntax or implied by the record type of focus line. (e.g. WHERE, LOCATE, etc.)

The current value DRECTYPE may be interrogated using the QUERY or EXTRACT DRECTYPE commands.

DB2 Table Display

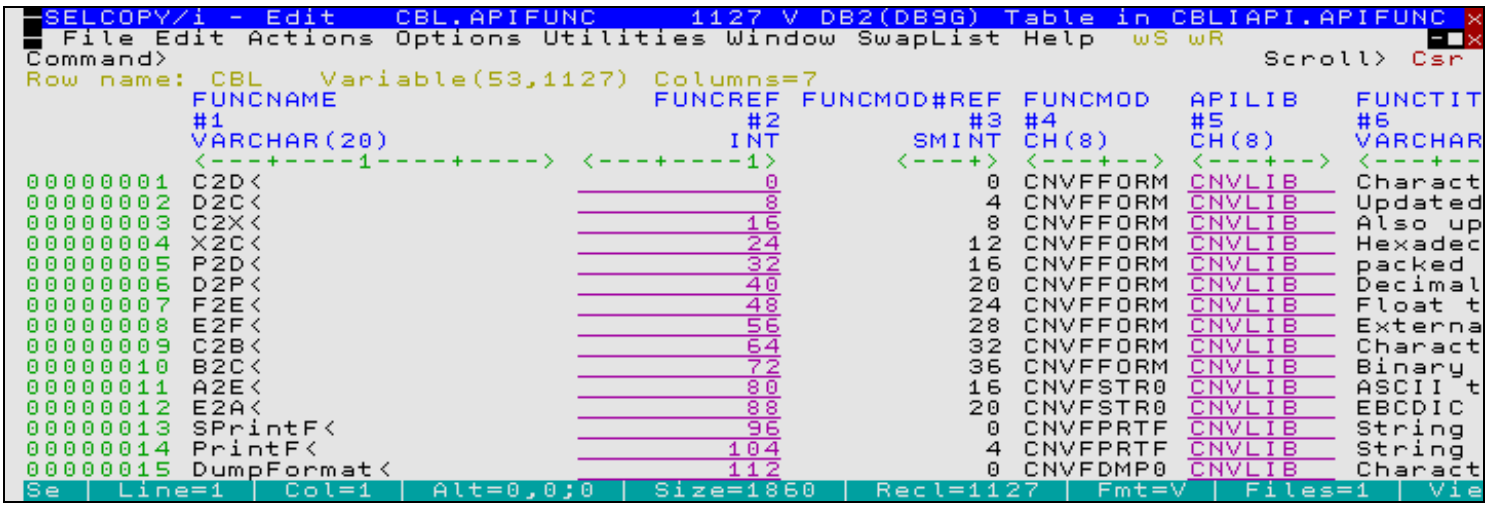

SELCOPY/i - Edit CBL.APIFUNC 1127 V DB2(DB9G) Table in CBLIAPI.APIFUNC
File Edit Actions Options Utilities Window SwapList Help wS wR
Command>
Scroll> Csr
Row name: CBL Variable(53,1127) Columns=7
FUNCNAME FUNCREF FUNCMOD#REF FUNCMOD APILIB FUNCTIT
#1 #2 #3 #4 #5 #6
VARCHAR(20) INT SMINT CH(8) CH(8) VARCHAR
00000001 C2D< 0 0 CNVFFORM CNVLIB Charact
00000002 D2C< 8 4 CNVFFORM CNVLIB Updated
00000003 C2X< 16 8 CNVFFORM CNVLIB Also up
00000004 X2C< 24 12 CNVFFORM CNVLIB Hexadec
00000005 P2D< 32 16 CNVFFORM CNVLIB packed
00000006 D2P< 40 20 CNVFFORM CNVLIB Decimal
00000007 F2E< 48 24 CNVFFORM CNVLIB Float t
00000008 E2F< 56 28 CNVFFORM CNVLIB Externa
00000009 C2B< 64 32 CNVFFORM CNVLIB Charact
00000010 B2C< 72 36 CNVFFORM CNVLIB Binary
00000011 A2E< 80 16 CNVFSTR0 CNVLIB ASCII t
00000012 E2A< 88 20 CNVFSTR0 CNVLIB EBCDIC
00000013 SPrintF< 96 0 CNVFPRTF CNVLIB String
00000014 PrintF< 104 4 CNVFPRTF CNVLIB String
00000015 DumpFormat< 112 0 CNVFDMP0 CNVLIB Charact
Se | Line=1 | Col=1 | Alt=0,0;0 | Size=1860 | Recl=1127 | Fmt=V | Files=1 | Vie

Figure 30. SDE DB2 Table Edit View.

Overview

Included as part of SELCOPY/i DB2 functionality, the SDE data editor includes support for BROWSE and EDIT of DB2 results tables. Using the SDE editor for DB2 table edit has the benefit that all SDE commands and operations, including FIND, CHANGE, WHERE and LOCATE, apply equally to the display and edit of DB2 table data.

For both BROWSE and EDIT, all rows of the DB2 results table are loaded into storage before being displayed in the SDE window view.

The SDO structure used to format the DB2 table rows may be a pre-defined structure file, generated using the CREATE STRUCTURE command, or, if no SDO structure file is named on the BROWSE or EDIT operation, a SELCOPY/i generated temporary structure derived from the DB2 SQLDA chain returned by SQL SELECT query.

A DB2 table edit SDO includes the SQL query option criteria (SELECT, WHERE, ORDER BY) used to obtain the results table, Isolation clause options specified on edit and other options which include skipping display of locked rows, actions that trigger an SQL COMMIT and preservation of primary key fields.

Because data in rows returned by a DB2 results table are formatted identically, the display of table rows has only one record type assigned a name (Row name) equal to the table owner id. Because there is only one record type, all table rows are eligible for processing by commands that operate on records of the default record type.

A separate SELCOPY/i DB2 table EDIT audit log may be started for each table edit session. This isolates SQL insert, delete and update operations performed against an individual results tables from other SQL operations performed in the same SELCOPY/i session. Note that multiple DB2 tables may be edited in the same SELCOPY/i session, potentially from different DB2 subsystems. See description of the Audit Trail Functions panel for details of SELCOPY/i DB2 auditing.

DB2 Table Edit/Browse

Differences between the SDE window display of structured data set records and that of DB2 tables, are detailed below.

Header Lines

The Record Type header line displays the row (record type) name assigned to the table rows and the number of columns defined in the results table.

The Field Type header line displays the the DB2 built-in data type as specified in the results table column definition. i.e. no field position is displayed and any applicable field length or field precision and scale are represented in "( )" (parentheses).

See *"SDE Data Types"* for descriptions of each of the supported data types and their representation in the Field Type header line.

Window View Format

DB2 table SDE views support multi record and single record views as displayed for parameters TABLE and SINGLE of the FORMAT command. Display of table data in unformatted character or hex (FORMAT CHAR/HEX) is not sensible. However, HEX ON/OFF may be used to display, not only the interpreted printable character representation of the formatted field, but also its raw hexadecimal data.

Record Information Columns

Record information (RECINFO) field ID is not applicable to DB2 table rows and so is always suppressed. However, record information field **SQLCODE** may be displayed to report the last SQL code received following an SQL INSERT, DELETE or UPDATE performed on the table row as a result of a SAVE operation.

DB2 Column Attributes

Single record view of a table row includes an additional column within the display which flags attributes defined to the individual table columns. These column attributes are obtained from the DB2 catalog table SYSIBM.SYSCOLUMNS and flags whether a column is part of the primary key, a unique key, foreign key, a duplicate key index, has a check constraint, a default value and/or supports a null value.

Display of the column attributes information may be suppressed using the COLATTRIBUTES ON/OFF option.

Varible Length Fields

Data in variable length character fields (VARCHAR) that is not of the maximum length is suffixed with a "<" (less than) character. The "<" itself does not form part of the data but is displayed to indicate that the character to the left is the last character of the field data. This indication is particularly necessary where the variable length field contains trailing blank characters which are to be preserved.

In editing the field data the "<" indicator may be overtyped or shifted right as appropriate. If overtyped or deleted, and the field data is not of the maximum length, then SELCOPY/i will reinsert the "<" indicator following the last **non-blank** character. (i.e. any trailing blanks will not be preserved.)

The "<" indicator may also be manually deleted and re-entered in blank data following the last non-blank character of the field. This technique may be used to insert trailing blanks in the field. Beware that, when using this technique, if the original "<" indicator is not deleted first then it will become part of the field data.

Pop-up Utility Menu

Pressing <PF4> in an SDE DB2 table edit view now opens the DB2 Browse/Edit Utilities Menu for DB2 specific table editing options instead of the SDE Browse/Edit Utilities Menu for structured file editing options.

Segmented Records

Records may be organised in such a way that they are split into a number of record segments, each segment being mapped by a unique structure (COBOL group or PL1 major/minor structure).

Segmented records begin with a single primary (base) segment immediately followed by any number of non-overlapping, secondary segments. A secondary segment may have the same or different segment record-type (RTO) mapping as other secondary segments in the record.

```
Record: 1
 +-----------+--------------+--------------+--------------+--------------+
 | Primary_1 | Secondary_1  | Secondary_1  | Secondary_1  | Secondary_1  |
 +-----------+--------------+--------------+--------------+--------------+

Record: 2
 +-----------+--------------+-----------------+-----------------+
 | Primary_1 | Secondary_1  | Secondary_2     | Secondary_2     |
 +-----------+--------------+-----------------+-----------------+

Record: 3
 +----------------+--------------+-------------------------------------------+
 | Primary_2      | Secondary_1  | Secondary_4                               |
 +----------------+--------------+-------------------------------------------+

Record: 4
 +-----------+--------------+-----------------+-----------------------+
 | Primary_1 | Secondary_1  | Secondary_2     | Secondary_3           |
 +-----------+--------------+-----------------+-----------------------+
```

The record data must contain ID fields that identify which segment mapping is to be used to format individual segments of the record.

ID fields that identify a primary segment mapping must exist within the primary segment data. ID fields that identify a secondary segment mapping may exist within the secondary segment data, within the data of any previously mapped segment belonging to the same record, or, specifically, within the primary segment data.

Structure Definition and Record-Type Assignment

A SELCOPY/i SDE structure may be created using the CREATE STRUCTURE command or Create Structure (SDO) Menu panels (=9) to format segmented records so that a record type (RTO) mapping definition exists for each record segment in the file.

Each RTO is defined as being either a primary (PRI) or secondary (SEC) segment map, or the default (primary) segment map (DEF). Except for the default segment RTO, which must **not** be defined with segment identification criteria, all primary and secondary segment RTOs must be defined with (USE WHEN) segment identification criteria.

A primary segment RTO is assigned to primary (base) segment of the record. However, if ID fields do not satisfy the segment identification criteria for any of the primary segment RTOs, the default segment RTO is used. See *"Record Type Assignment"* which defines the precedence for assigning primary segment RTOs.

Unprocessed record data that follows the primary segment is assigned a secondary segment RTO based on associated identification criteria. The assignment of secondary segment RTOs is repeated for all remaining record data until the end of record is reached or the remaining data does not satisfy any segment identification criteria belonging to the secondary segment RTOs. In the latter case, the remaining record data is displayed as unformatted character data with the record type "UnmappedSeg".

Specifying USE WHEN Segment Identification Criteria

USE WHEN segment identification criteria is mandatory for non-default primary and secondary segment RTO definitions.

Segment identification criteria associated with each RTO definition is based on an SDE expression. This identifies all ID fields within the record data and also values against which those ID fields will be tested in order to determine whether the RTO segment mapping is suitable to be assigned to an individual record segment.

Within any RTO segment mapping USE WHEN expression, ID fields may be referenced using the following methods:

Unformatted data position and length

Built-in function SEGPOS() may be used in the USE WHEN *expression* record type identification criteria to reference a data field in the current segment or the segment immediately preceding the current segment. Similarly, BASPOS() may be used to reference a data field in the primary (base) segment only.

e.g. The following identification criterion is based on an ID field of length 3 characters located 10 bytes before the end of the segment immediately preceding the current segment:

```
USE WHEN  SegPos(-10,3) = C'X12'
```

If the ID field is a Packed Decimal value then LEFTDEC() may be used to obtain the numeric value with or without a scale. e.g. The following ID field is a packed decimal field located at position 12 of the primary segment:

```
USE WHEN  LeftDec(BasPos(12)) > 31
```

Formatted data field

A formatted ID field references a field in the current segment or within a previous segment of a specific structure (record type) name. If referencing a field in a previous segment, the record type name which maps that segment must be specified as a high level qualifier to the field name (i.e. *record-type.field_name*). If more than one preceding segment is mapped using *record-type*, then the the one closest to the current segment is tested.

Because formatted fields have a specific data type, the USE WHEN expression may involve operators and terms based on the data type of the ID field. e.g. The following identification criterion is based on a floating point ID field XVAL in a preceding segment of record type MAPBASE1:

```
USE WHEN  MAPBASE1.XVAL > 27.83
```

Segmented Records SDE Browse/Edit View

As for display of non-segmented records, files containing segmented records may be displayed in multi-record (FORMAT TABLE/CHAR/HEX) or single-record (FORMAT SINGLE) view.

Primary commands CHAR, MAP (FMT), VFMT and UNFMT are supported to display the current segment in multi-record unformatted character view, single-record formatted view, multi-record formatted view and single-record unformatted character view respectively.

Data in record segments is displayed in exactly the same way as non-segmented records. Prefix area, record type headers and record information is also displayed as for non-segmented records with the following exceptions:

1. Segmented record header display differs from that of non-segmented records only in that, instead of "**Record Type:**", header line 1 of each group of matching segment types displays either "**Base:**", for primary segments; "**Base(D):**", for

default primary segments; or "**Segment:**", for secondary segments.

In order to easily distinguish between primary and secondary segments, the header line 1 of secondary segments is coloured yellow by default. See SET/QUERY/EXTRACT option COLOUR (or COLOR) SEGMENT.

2. For secondary segments only, the record information id column, displayed using the SET/QUERY/EXTRACT option RECINFO, displays the offset into the record of the start of the segment.

   Primary segments display the TTR/Offset or RBA value of the start of the record as for non-segmented records.

3. Prefix area numbers no longer correspond to record numbers within the edited file but to the sequence of segments. Therefore, navigating the file data is performed by segment and not record number. e.g. "LOCATE 32" will scroll to the 35th (primary or secondary) segment, not the 35th record, of the file.

Navigating Segmented Record Display

In both multi-record and single-record views, the display of segments may be navigated using standard SDE CLI (primary) commands FIND, LOCATE, UP, DOWN, LEFT, RIGHT, TOP, BOTTOM, NEXT and PREV.

Note that, in single-record view, LEFT and RIGHT will scroll backwards and forwards respectively through every segment in the file regardless of the type of segment (primary or secondary) or record to which a segment belongs.

Commands NEXT and PREV may be used to restict scrolling to only secondary segments belonging to the current record. This is the NEXT and PREV default, however, parameters are supported to scroll to the next and previous primary or unmapped (UnmappedSeg) segments.

Segmented Record Edit

SELCOPY/i SDE segmented record edit is supported for all file organisations other than VSAM KSDS.

Full Edit, Auxiliary Edit and Update-in-Place Edit SDE edit techniques are all supported for segmented record edit.

Data Updates that affect Record Segment Mapping

The RTO mapping assigned to a segment is determined by values in ID fields which may occur in the same segment or within the primary and/or previous secondary segments. Therefore, any change to a segment may affect the RTO mapping identification criteria for that segment or any subsequent segment in the record.

SELCOPY/i SDE is able to detect changes to ID fields which have been identified via a USE WHEN expression that includes only fields referenced explicity by name or field reference number. Changes to ID field name "Unmapped" (reference number #1) are detected only when record-type formatting is disabled. Where an ID field change is detected, or where the length of the segment data is altered via CHANGE or RECLEN updates, the ID flag is set on for the changed segment.

The ID flag, represented by "m" in the record information flag display and ==ID> in multi-record view prefix area, is an indication that the record to which this segment belongs may require a remap (re-assignment of segment RTOs) based on new ID field values. This flag may be ignored but may be of importance if the user wants to first verify that integrity between the required segment mapping and its identification criteria is maintained prior to saving the data and ending the edit session.

Remap of a record and reset of the ID flag is achieved using the primary command IDENTIFY, or prefix (line) commands **ID*<n>*** or **IDD**. Note that the remap is **not** performed automatically in order to allow the user the opportunity to make further changes to the data before executing IDENTIFY. Note that, if ID field values do not satisfy any secondary segment mapping identification critera, the remainder of that record is displayed as unformatted data (UnmappedSeg).

Remap of a record may also be required where an update has occurred for an ID field identified in a USE WHEN expression by its unformatted record position and length, or when segments have been inserted, deleted, duplicated, copied or moved. For this reason, the default action of the IDENTIFY command is to not only remap records containing segments with the ID flag, but also records flagged as having changed segments. (See option IDSCOPE to force IDENTIFY to remap **only** those records containing a segment with ID flag set on).

Full Edit Functionality

Update-in-place edit only supports changing data in existing segments without changing the the length of a segment. In contrast, **full function** edit supports changing data, changing the length of a segment and also the addition and removal of record segments.

When segments are displayed in multi-record view, individual segments (or blocks of consecutive segments) may be replicated, deleted, inserted, moved or copied using standard SDE edit primary and line commands.

In single record view, segments may only be deleted, duplicated or inserted using primary commands DELETE, DUPLICATE and INSERT respectively.

Edit actions that affect the location of a primary segment within the file directly affects the location of the record to which it belongs. In contrast, secondary segments may be deleted, moved, copied, duplicated and inserted without affecting the location of a record within the file. For secondary segments, these edit operations may only increase or decrease the length of existing records.

Record Truncation

Beware that full edit operations involving primary segments can potentially split and join together records in the file.

If as a result of adding secondary segments to a record, the length of that record exceeds the maximum defined logical record length of the file (LRECL), then remapping the record or executing a save operation will truncate the excess data. For variable record format files, if data at the end of the truncated record represents an incomplete record segment map, then that data is also truncated.

Following an execution of IDENTIFY in which record truncation occurs, the truncated segments are removed from the display, the truncation error flag (**=TRNC>**) is set on in the primary segment of each truncated record and the following message is returned:

```
ZZSD447I IDENTIFY process caused 1 physical records to be truncated
```

The UNDO command may be executed to undo the remap and recover the truncated segments. The user may then correct the cause of the truncation.

Following execution of SAVE, FILE or END in which record truncation occurs, the truncated segments are **not** removed from the display, the truncation error flag (**=TRNC>**) is set on in the primary segment of every truncated record and the following message is returned:

```
ZZSD406E The physical record associated with a base segment is too long
     (greater than nnn). Truncated to length mmm.
```

Where *nnn* is the maximum allowable record length and *mmm* is the length at which the record was truncated. If truncation occurs on execution of FILE or END, the close of the SDE edit view is temporarily disabled in order to allow the user the opportunity to correct the truncation error.

Although the file has been saved to disk with truncated records, the in-storage copy of the records remains unchanged. Therefore, the user may correct and re-save the file data before closing the edit view.

Full Edit Operations

The effects of full edit operations that add, remove and/or reposition record segments are summarised below:

DELETE

Delete individual segments using command DELETE or prefix (line) command **D<*n*>** (where *n* is the number of segments to be deleted). Alternatively, delete a block of segments using prefix command **DD** on the first and last segments to be deleted.

On deleting a primary (base) segment, the record to which the primary segment belongs is deleted and its secondary segments are appended to segments belonging to the previous record.

Deleting a secondary segment simply removes that segment from the record, hence reducing the record length. All subsequent segments in the record are shifted left to fill the gap left by the deleted segment.

INSERT

Insert individual segments following the focus segment using command INSERT or prefix (line) command **I<*n*>** (where *n* is the number of segments of the focus segment type to be inserted).

Inserting a new primary segment inserts a new record in the file. If the inserted primary segment is inserted before a secondary segment (i.e. in the middle of an existing record), then the existing record is split so that all secondary segments following the inserted primary segment now belong to the new record.

Inserting a new secondary segment simply adds the segment to the focus record.

COPY and **DUPLICATE**

Copy individual segments using prefix (line) command **C<*n*>** (where *n* is the number of segments to be copied). Alternatively, copy a block of segments using prefix command **CC** on the first and last segments to be copied. Prefix commands **A** or **B** must be used to specify whether segments are to be copied After or Before the selected segment.

Duplicate a segment using command DUPLICATE or prefix (line) command **R<*n*>** (where *n* is the number times the segment is to be duplicated). Alternatively, duplicate a block of segments using prefix command **RR<*n*>** on the first and last segments to be duplicated.

The effect of a copy or duplicate operation on existing primary and secondary segments is the same as for insert.

MOVE

Move individual segments using prefix (line) command **M<*n*>** (where *n* is the number of segments to be moved). Alternatively, move a block of segments using prefix command **MM** on the first and last segments to be moved. Prefix commands **A** or **B** must be used to specify whether segments are to be moved After or Before the selected segment.

The effect of a move operation on existing primary and secondary segments located at the destination of the move is the same as for insert.

The effect of a move operation on primary and secondary segments immediately preceding and following the original location of the moved segments is the same as for delete.

Expressions

A selection of SDE commands support an expression as parameter input. e.g. The Edit and Browse FILTER parameter, LOCATE, WHERE and USE.

An expression consists of one or more terms (literal strings, numerical values, field values, function calls or sub-expressions) and zero or more operators. e.g.

```
term1

term1  operator1  term2  operator2  term3

operator1  term1
```

An expression is evaluated from left to right with modifications imposed by the presence of sub-expressions and the defined precedence of operators. Each term within an expression is evaluated as appropriate.

The result of an expression evaluation is either a numerical value or a character string. If numerical, then the result is considered to have a Boolean value whereby a zero (0) result has the value "false" and any non-zero result has the value "true".

Expression Terms

Individual terms within an expression are evaluated as they are encountered parsing from left to right. Following evaluation of a term, the resultant value is assigned a data type which is stored internally by the program.

A user need not know the internal data type of an evaluated term, simply that it is either character or numerical. For details on specific data types used by SDE, see SDE Data Types.

A term may be one of the following:

Literal string

A literal string stored internally as a character data type and is represented as a delimited string, character string or a hexadecimal string. A literal string may be null (length zero). e.g. C''.

Delimited string

A string of character text, possibly including blanks and special characters, enclosed in quotation marks (") or apostrophes ('). Alphabetic characters in a delimited string are treated as being both upper and lower case. Therefore, the following are considered equal:

```
'TERM XX' = 'term xx' = 'Term xx' = "tErm Xx"
```

Character string

A string of character text in quotation marks (") or apostrophes (') and prefixed with "C" or "c". The case of alphabetic characters in a delimited string are respected. e.g. C'TERMXX' is not equal to C'Termxx'.

Hexadecimal string

A string of even length consisting of valid hexadecimal characters (0-9, A-F, a-f) in quotation marks (") or apostrophes (') and prefixed with "X" or "x". e.g. X'E3859994F1' is equal to EBCDIC C'Term1'.

Numerical value

A numeric value is an unquoted signed integer or rational number. The decimal point in a rational number is represented by a period (.) and non-significant zeroes are also permitted. The sign of a numerical value is represented by one of the prefix operators, unary plus or unary minus. e.g. 12, +19, +0027.00, 67.353, -0.30

A numerical value is assigned a specific numeric data type based on its value. This data type may be converted during the evaluation of the expression in order to satisfy arithmetic and relational operations.

Field value

A field value is either a hash (#) prefixed number or an unquoted character string that matches a formatted field column reference number or field column name belonging to the named (or focus) record type.

A field name may be fully qualified, partially qualified or unqualified. An unqualified field name is simply the field's designated name (e.g. ADDR), a partially qualified field name includes the names of any nested groups in which the field is defined (e.g. MAST.ADDR or INVOICE.ADDR) and a fully qualified field name includes the record type name (e.g. LOCATION-REC.MAST.ADDR).

Note that the default qualifier separator character is "." (dot) but may be altered using the QSEPARATOR option. Furthermore, if the ABBREVIATION option is set on, then any group or field item designator within the file name may be abbreviated, starting with the first letter of the designator, to a length that identifies it as a unique group name or field item.

The data type of the field value matches that of the field being referenced so that a field which is of numerical data type will evaluate to be a numerical value and a field which is of character data type will evaluate to be a literal string with character case preserved.

A field column reference/name may be immediately followed by a subscript in a left and right parenthesis. This references the value of an individual entry within a field array. If the field array contains multiple dimensions, then a subscript must exist for each dimension of the array in order to identify a single field value. e.g. #12(5), RoomSize(6,2,4).

The nominated field is evaluated to be that field's value within the next data record assigned the identified record type. The data type assigned to the evaluated value is that of the nominated field.

If record data is unformatted, the record data belongs to a single field of field name "UnMapped" or, for unformatted record segments, "UnMappedSeg" and field reference number #1.

To reference record data in its unformatted state (whether or not a record type mapping is assigned), the buit-in function RECORD() may be used.

Function Call

A function call is a call to one of SELCOPY/i's internal routines which return a single result string or numerical value. The function name is a an unquoted character string immediately followed by zero or more, comma separated function arguments bracketed by a left and right parenthesis.

The data type of the value returned from a function call will be that defined by the function. See Function Calls for all available functions supported by the SELCOPY/i SDE environment and their return values.

Sub-expression

A sub-expression is any expression which is bracketed by a left and right parenthesis.

Expression Operators

An operator may act on the pair of terms between which it is positioned or, in the case of unary plus, minus and logical NOT which are prefix operators, act on the single term immediately to the right of the operator.

Operators may be divided into the following 3 categories: Arithmetic, Relational and Logical.

Arithmetic Operators

Multiple numerical values, numerical field values and function calls that yield a numerical value, may be combined using arithmetic operators so resulting in another numerical value when evaluated.

The data type of an evaluated term acted upon by the arithmetic operator, must be numerical otherwise error ZZSD068E is returned.

In order to perform the arithmetic operation, the numerical data type one of the terms involved may be temporarily converted to match the data type of the other term so as to preserve numerical precision and scale. This numerical data type is then assigned to the resultant value.

Supported arithmetic operators are:

Operator	Description
+	Add.
-	Subtract.
Prefix +	Unary plus. Equivalent to: "0+number".
Prefix -	Unary minus. Equivalent to: "0-number".

Relational Operators

The value returned on evaluating a relational operator is either "1" if the result is "true", or "0" if the value is "false".

If the terms acted upon by a relational operator are numerical, then the comparison will be arithmetic and signed. In this case the numerical data type of one or both of the terms may be temporarily converted before performing the evaluation.

If one of the terms has a character data type, then the comparison will be logical so that a simple character-by-character compare occurs. Except for relational operators ">>", "<<", "\>>" and "\<<", the shorter string is padded on the right with blanks.

Supported relational operators are:

Operator	Description
=	Equal.
\= ¬= != <> **NOT=**	Not equal, less than or greater than.
>	Greater than.
\> ¬> !> <= **NOT>**	Not greater than, less than or equal.
<	Less than.
\< ¬< !< >= **NOT<**	Not less than, greater than or equal.
>>	Beginning. (For logical compare only.)
\>> ¬>> !>> **NOT>>**	Not beginning. (For logical compare only.)
<<	Contains. (For logical compare only.)
\<< ¬<< !<< **NOT<<**	Does not contain. (For logical compare only.)

Logical Operators

Logical operators are Boolean and so, like relational operators, return a value of "1" for "true" and "0" for "false".

The terms acted upon by a logical operator must evaluate to numerical values otherwise error ZZSD068E is returned. Any non-zero value is treated as being "1", true.

Supported logical operators are:

Operator	Description
& AND	And. Returns 1 if both terms are true.
| ¦ OR	Inclusive Or. Returns 1 if either term is true.
Prefix ¬ Prefix ** **Prefix NOT	Logical Not. Negates the term so that 0 is returned if the term is true, 1 if the term is false.

Parentheses and Operator Precedence

An expression and individual terms within an expression are evaluated from left to right. However, the order in which operators are actioned depends on their precedence and the presence of parentheses.

An operator with a higher precedence will be actioned before operators with a lower precedence. This process is repeated until the entire expression is evaluated. e.g. In the following expression, where operator2 has a higher precedence than operator1:

```
term1  operator1  term2  operator2  term3
```

The sub-expression `(term2 operator2 term3)` will be evaluated first.

When parentheses are encountered, the entire sub-expression between the parentheses is evaluated immediately when the term is required. In this way parentheses may be used to force the action of an operator with a lower precedence before that with a higher precedence. e.g. Logical Not has a higher precedence than logical And, therefore `¬1&0` evaluates to 0, however `¬(1&0)` evaluates to 1.

The order of operator precedence is as follows (highest at the top):

Prefix operators	`Prefix ¬  Prefix +  Prefix -`
Arithmetic operators Add and Subtract	`+  -`
Relational operators	`=  \=  >  <  >=  <=  <<  \<<  >>  \>>`

Logical operator And	&
Logical operator Or	|

Function Calls

Functions are internal routines that return a single result string or value, and may be included as a term within an expression.

A function call is referenced by the name of the function immediately followed by zero or more, comma separated argument expressions enclosed in parentheses. Each argument expression may itself contain function calls.

If the function has zero arguments, then the parentheses may be omitted. e.g. function call LRECL() may be expressed simply as LRECL. However, specification of parentheses will distinguish the function from a non-subscripted record field with the same name. If such a field exists in the focus record type and no parentheses are specified following the function name, then the field name is used for evaluation.

Note that, when parentheses are specified, there can be no intervening blanks between the function name and the opening parenthesis.

Subscripted Field Name Conflict

Subscripted field names are used to reference array elements within a field belonging to the focus record type. e.g. RoomDimensions(5,1,2)

Because function calls are expressed in the same way as subscripted fields, a conflict may arise when attempting to reference a function call which has the same name as a field containing array elements.

If the function name matches the name of a subscripted field, then the function is not called and the field is evaluated instead. To force use of the function name, the function's internal name should be used. This is the name of the function prefixed with "BIF". e.g. The internal name of the FIND() function is BIFFIND().

ABBREV()

```
>>-- ABBREV( --- string1 -- , --- string2 ---+------------------+-- ) --------><
                                              |                  |
                                              +-- , -- length --+
```

ABBREV() or BIFABBREV() returns "1" if *string2* is equal to the leading characters of *string1* and *string2* has a length that is greater than or equal to *length*, otherwise "0" is returned.

If not specified, *length* defaults to the length of *string2*.

Note that the data type of both *string1* and *string2* must be character and the data type of *length* must be positive integer otherwise ZZSD386E is returned.

Examples:

```
ABBREV('input','inp')     =  1
ABBREV(c'input','inp')    =  0
ABBREV('input','inp',4)   =  1
ABBREV('input','inp',2)   =  0
ABBREV('input','int')     =  0
```

BASPOS()

```
>>-- BASPOS( --- start -+--------------------+- ) ----------------------------><
                        |                    |
                        +- , --- length ----+
```

BASPOS() or BIFBASPOS() returns a character string which is a sub-string of the current record's primary (base) segment starting at position *start* for length *length*. If the current structure is not segmented, then the current record is used as the primary segment.

BASPOS() is primarily used for mapping segmented records where a record-type definition (RTO) is assigned to secondary segments based on field values in the primary segment.

If *start* is greater than the length of the primary segment, then a blank character string is returned. If *length* is greater than the length of the primary segment, the returned character string will include data from the start of the segment that follows the primary segment.

If not specified, *length* defaults to be the length of the primary segment from position *start* to the end of the primary segment.

Note that the data type of the primary segment is character and *start* and *length* must be positive integer values otherwise ZZSD386E is returned.

Examples:

In the following, the primary segment contains a 2 character country code `c'UK'` at position 15.

```
BASPOS(15,2)                =  'UK'
```

CHANGED()

```
>>-- CHANGED() ------------------------------------------------------------><
```

CHANGED() or BIFCHANGED() has no arguments. It returns "1" if the focus record, segment or DB2 table row has been changed in the current edit session and "0" otherwise.

Because CHANGED() has no arguments, it may be expressed simply as CHANGED.

CHANGEERROR()

```
>>-- CHANGEERROR() ---------------------------------------------------------><
```

CHANGEERROR() or BIFCHANGEERROR() has no arguments. It returns "1" if a CHANGE command error (**==ERR>**) has been flagged on the focus record, segment or DB2 table row in the current edit session and "0" otherwise.

Because CHANGEERROR() has no arguments, it may be expressed simply as CHANGEERROR.

CHANGEOK()

```
>>-- CHANGEOK() ------------------------------------------------------------><
```

CHANGEOK() or BIFCHANGEOK() has no arguments. It returns "1" if a CHANGE command executed successfully (**==CHG>**) on the focus record, segment or DB2 table row in the current edit session and "0" otherwise.

Because CHANGEOK() has no arguments, it may be expressed simply as CHANGEOK.

DATATYPEERROR()

```
>>-- DATATYPEERROR() -------------------------------------------------------><
```

DATATYPEERROR() or BIFDATATYPEERROR() has no arguments. It returns "1" if the focus record or segment is assigned a non-default record type and one or more of its fields contains data invalid for the field's data type (e.g. invalid packed decimal data in a DECIMAL field) and "0" otherwise.

Because DATATYPEERROR() has no arguments, it may be expressed simply as DATATYPEERROR.

DUPLICATEKEY()

```
>>-- DUPLICATEKEY() ------------------------------------------------------><
```

DUPLICATEKEY() or BIFDUPLICATEKEY() has no arguments. It applies to KSDS edit mode only and returns "1" when either of the following is true, otherwise "0" is returned:

1. Following a SAVE operation, the focus record was flagged as having a duplicate key with another record which has been changed or inserted in the current edit session (**=DUPE>**).
2. Following a SAVE operation, the focus record was flagged as having a duplicate key with another unchanged record whithin the file (**=DUPF>**).

Because DUPLICATEKEY() has no arguments, it may be expressed simply as DUPLICATEKEY.

EMPTYSLOT()

```
>>-- EMPTYSLOT() ---------------------------------------------------------><
```

EMPTYSLOT() or BIFEMPTYSLOT() has no arguments. It applies to edit of VSAM RRDS and VRDS only and returns "1" if, on load, the focus record was flagged as being an empty slot and "0" otherwise.

Because EMPTYSLOT() has no arguments, it may be expressed simply as EMPTYSLOT.

EXCLUDED()

```
>>-- EXCLUDED() ----------------------------------------------------------><
```

EXCLUDED() or BIFEXCLUDED() has no arguments. It returns "1" if the focus record, segment or DB2 table row belongs to a group of one or more excluded lines in the current edit session and "0" otherwise.

Because EXCLUDED() has no arguments, it may be expressed simply as EXCLUDED.

FIND()

```
>>-- FIND( -- find_syntax ----- ) ----------------------------------------><
```

FIND() or BIFFIND() supports a single argument *find_syntax* which is a literal or charcter string comprising a structured data edit FIND command parameter string. Note that the parameter string will be upper cased unless specified as a character string (i.e. specified in quotation marks (") or apostrophes (') and prefixed with "c" or "C".

If successful, FIND() returns the position of the found string within the first data field that contains a match for the search string, otherwise zero is returned. If no structure is applied to the data, this position will be a position in the unexpanded record.

If the search string is a numerical value which is found in a field of numerical data type, then the returned value is "1".

Examples:

```
FIND( c" c'Needle' 21 50 " )
FIND( " 'hit' last prefix (#2:#4) " )
FIND( " 22  (#8,#12)" )
```

HASPOINT()

```
>>-- HASPOINT() ------------------------------------------------------------><
```

HASPOINT() or BIFHASPOINT() has no arguments. It returns "1" if the focus record, segment or DB2 table row has an active SET POINT label name and "0" otherwise.

Because HASPOINT() has no arguments, it may be expressed simply as HASPOINT.

HASPREFIXCMD()

```
>>-- HASPREFIXCMD() ---------------------------------------------------------><
```

HASPREFIXCMD() or BIFHASPREFIXCMD() has no arguments. It returns "1" if the focus record, segment or DB2 table row has a prefix command pending on the line and "0" otherwise.

Because HASPREFIXCMD() has no arguments, it may be expressed simply as HASPREFIXCMD.

IDREQUIRED()

```
>>-- IDREQUIRED() ----------------------------------------------------------><
```

IDREQUIRED() or BIFIDREQUIRED() has no arguments. It returns "1" if an ID field (identified by USE WHEN criteria) belonging to the focus record or segment, has been changed and so flagged as possibly requiring execution of the IDENTIFY command (**==ID>**) to remap the record. Otherwise "0" is returned.

Because IDREQUIRED() has no arguments, it may be expressed simply as IDREQUIRED.

INSERTED()

```
>>-- INSERTED() ------------------------------------------------------------><
```

INSERTED() or BIFINSERTED() has no arguments. It returns "1" if the focus record, segment or DB2 table row has been inserted in the current edit session and "0" otherwise.

Because INSERTED() has no arguments, it may be expressed simply as INSERTED.

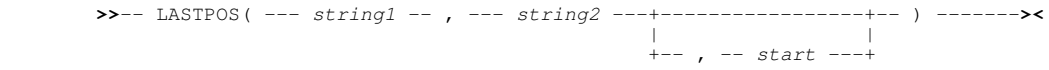
KEYCHANGED()

```
>>-- KEYCHANGED() ----------------------------------------------------------><
```

KEYCHANGED() or BIFKEYCHANGED() has no arguments. It applies to KSDS edit mode only and returns "1" if the focus record is flagged as having had a change to its key during the current edit session and "0" otherwise.

Because KEYCHANGED() has no arguments, it may be expressed simply as KEYCHANGED.

LASTPOS()



LASTPOS() or BIFLASTPOS() returns the position of the last occurrence of *string1* in *string2* starting the backwards search from position *start* in *string2*. A search is successful only if *string1* exists entirely between position 1 of *string2* and the *start* position. If *string1* is not found, "0" is returned.

If not specified or a value greater than the length of *string2*, *start* defaults to be the last position (i.e. length) of *string2*.

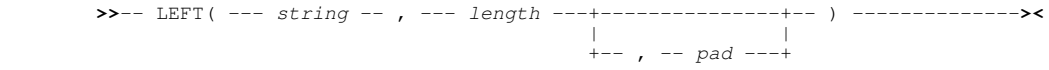
Note that the data type of both *string1* and *string2* must be character and the data type of *start* must be a positive integer otherwise ZZSD386E is returned.

Examples:

In the following, field name SourceText exists in the focus record type and contains character data `c'To be or not to be'`.

```
LASTPOS('be',SourceText)       =  0
LASTPOS(c'be',SourceText)      =  17
LASTPOS(c'be',SourceText,17)   =  4
```

LEFT()



LEFT() or BIFLEFT() returns a character string equal to the left-most *length* characters of *string*. If *length* is greater than the length of *string*, the returned character string is padded on the right with the *pad* character.

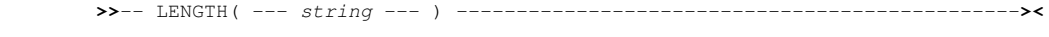
If not specified, *pad* defaults to be character blank.

Note that the data type of *string1* must be character, *length* must be a positive integer and *pad* must be a character string of length 1 otherwise ZZSD386E is returned.

Examples:

```
LEFT('Hello World',5)       =  'HELLO'
LEFT(c'Hello World',15)     =  'Hello World    '
LEFT(c'Hello World',13,'#') =  'Hello World##'
```

LENGTH()



LENGTH() or BIFLENGTH() returns the length of *string*.

Note that the data type of *string* must be character otherwise ZZSD386E is returned.

Examples:

In the following, field reference #13 contains character data `c'James Joyce'`.

```
LENGTH('Hello')    =  5
LENGTH(#13)        =  11
```

LENGTHERROR()

```
>>-- LENGTHERROR() ------------------------------------------------------><
```

LENGTHERROR() or BIFLENGTHERROR() has no arguments. It returns "1" if the focus record or segment is flagged as having been assigned a record-type whereby the length of the record/segment falls outside the limits of possible lengths for the record type (**=LGTH>**). Otherwise "0" is returned.

Because LENGTHERROR() has no arguments, it may be expressed simply as LENGTHERROR.

LOWER()

```
>>-- LOWER( --- string --- ) ---------------------------------------------><
```

LOWER() or BIFLOWER() returns the *string* with all upper case alpha characters lower cased and all other characters unchanged.

Note that the data type of *string* must be character otherwise ZZSD386E is returned.

Examples:

In the following, field array element Email(2) contains character data `c'Mark123@CBL.COM'`.

```
LOWER('HELLO')   =  'hello'
LOWER(email(2))  =  'mark123@cbl.com'
```

LRECL()

```
>>-- LRECL() ----------------------------------------------------------><
```

LRECL() or BIFLRECL() has no arguments. It returns the length of the focus record, segment or DB2 table row.

Because LRECL() has no arguments, it may be expressed simply as LRECL.

NOEOL()

```
>>-- NOEOL() ----------------------------------------------------------><
```

NOEOL() or BIFNOEOL() has no arguments and applies to HFS files containing record EOL characters. This function returns "1" if the focus record or segment is flagged as having been split due to no EOL delimiter being found within the file's declared LRECL length (**==EOL>**). i.e. record length is graeter than LRECL. Otherwise the function returns "0".

Because NOEOL() has no arguments, it may be expressed simply as NOEOL.

POS()

```
>>-- POS( --- string1 -- , --- string2 ---+------------------+-- ) -----------><
                                           |                  |
                                           +-- , -- start ---+
```

POS() or BIFPOS() returns the position of the first occurrence of *string1* in *string2* starting the search from position *start* in *string2*. A search is successful only if *string1* exists entirely between the *start* position and the last position of *string2*. If *string1* is not found,

"0" is returned.

If not specified, *start* defaults to be the first position of *string2*.

Note that the data type of both *string1* and *string2* must be character and the data type of *start* must be a positive integer otherwise ZZSD386E is returned.

Examples:

In the following, field name SourceText exists in the focus record type and contains character data `c'To be or not to be'`.

```
POS('be',SourceText)       =  0
POS(c'be',SourceText)      =  4
POS(c'be',SourceText,5)    =  17
POS(c' ',SourceText,10)    =  13
```

RECORD()

```
>>-- RECORD( ---+---------------------+-- ) ----------------------------------><
               |                     |
               +- relative_record# --+
```

RECORD() or BIFRECORD() returns a character string which is the unformatted contents of the record identified by *relative_record#*, a record number relative to the focus record.

If not specified, *relative_record#* defaults to be 0 (zero) identifying the focus record. A negative *relative_record#* value identifies a record occurring before the focus record, a positive value identifies a record occurring after the focus record. If the specified record does not exist the null string is returned.

Note that *relative_record#* must be zero, positive or negative integer value otherwise ZZSD386E is returned.

If RECORD() is specified with no arguments, it may be expressed simply as RECORD.

Examples:

```
RECORD()     * Contents of the focus record.
RECORD(-1)   * Contents of the record immediately before the focus record.
RECORD(2)    * Contents of the 2nd record following the focus record.
```

RIGHT()

```
>>-- RIGHT( --- string -- , --- length ---+---------------+-- ) -------------><
                                          |               |
                                          +-- , -- pad ---+
```

RIGHT() or BIFRIGHT() returns a character string equal to the right-most *length* characters of *string*. If *length* is greater than the length of *string*, the returned character string is padded on the left with the *pad* character.

If not specified, *pad* defaults to be character blank.

Note that the data type of *string1* must be character, *length* must be a positive integer and *pad* must be a character string of length 1 otherwise ZZSD386E is returned.

Examples:

In the following, field name XLen exists in the focus record type and contains binary integer value `8`.

```
RIGHT('Hello World',5)       =  'WORLD'
RIGHT(c'Hello World',15)     =  '    Hello World'
RIGHT(c'Hello World',13,'#') =  '##Hello World'
RIGHT(c'Hello World',XLen)   =  'lo World'
```

SAVEREQUIRED()

```
>>-- SAVEREQUIRED() ------------------------------------------------------><
```

SAVEREQUIRED() or BIFSAVEREQUIRED() has no arguments. It returns "1" if the focus record, segment or DB2 table row has been changed or inserted and has not yet been saved, and "0" otherwise.

Because SAVEREQUIRED() has no arguments, it may be expressed simply as SAVEREQUIRED.

SEGPOS()

```
>>-- SEGPOS( --- start -+--------------------+- ) ----------------------------><
                        |                    |
                        +- , --- length ----+
```

SEGPOS() or BIFSEGPOS() returns a character string which is a sub-string of the current or previous segment within the current record starting at position *start* for length *length*.

SEGPOS() is primarily used for mapping segmented records where a record-type definition (RTO) is assigned to segments based on field values in the current or previous segment.

If *start* is a negative value, then it refers to a position that many characters from the end of in the previous segment.

If *start* is greater than the length of the record, then a blank character string is returned. If *length* is greater than the length of the record, the returned character string is padded on the left with the blank character.

If not specified, *length* defaults to be the length of data from position *start* to the end of the segment in which *start* is located.

Note that the data type of the segment is character, *start* must be a positive or negative integer value and *length* must be a positive integer value otherwise ZZSD386E is returned.

Examples:

In the following, the segment previous to the segment currently being mapped contains the 11 character record-type name of the next segment starting 16 characters before the end of that segment (c'REC-CUST-01'). Similarly, the current segment data contains this name at position 31 of the segment.

```
SEGPOS(-16,11)              =  'REC-CUST-01'
SEGPOS(31,8)                =  'REC-CUST'
```

STRIP()

```
>>-- STRIP( --- string -- , --- option ---+-----------------+-- ) ------------><
                                          |                 |
                                          +-- , -- char ---+
```

STRIP() or BIFSTRIP() returns character string *string* with leading and/or trailing *char* characters removed.

The *option* argument may be specified as character string "L", "T" or "B" (upper or lower case) to indicate respectively that Leading, Trailing or Both leading and trailing characters are to be stripped. If not specified, *option* defaults to "B".

The *char* argument nominates the character to be stripped and, if not specified, defaults to be character blank. Each consecutive occurrence of *char* at the start and/or end of *string* will be stripped until a character other than *char* is encountered at which point the strip algorithm stops.

Note that the data type of *string1*, *option* and *char* must be character with *option* and *char* being of length 1, otherwise ZZSD386E is returned.

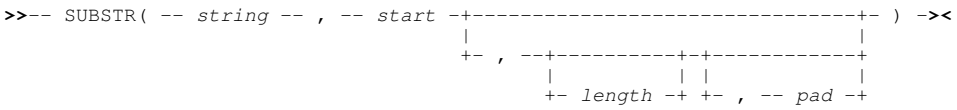
Examples:

In the following, field reference #2 in the focus record type contains character data `c' Time to go. '`.

```
STRIP('  Hello   World ')           =  'HELLO   WORLD'
```

```
STRIP(c'##CPU Seconds###','L','#')    =  'CPU Seconds###'
STRIP(c'##CPU Seconds###','t','#')    =  '##CPU Seconds'
STRIP(#2,'B')                         =  'Time  to  go.'
```

SUBSTR()



SUBSTR() or BIFSUBSTR() returns a character string which is a sub-string of *string* starting at position *start* within *string* for length *length*.

If *start* is greater than the length of *string*, then a blank character string is returned. If *length* is greater than the length of *string*, the returned character string is padded on the left with the *pad* character.

If not specified, *length* defaults to be the length of *string* from position *start* to the end of *string* and *pad* defaults to be character blank.

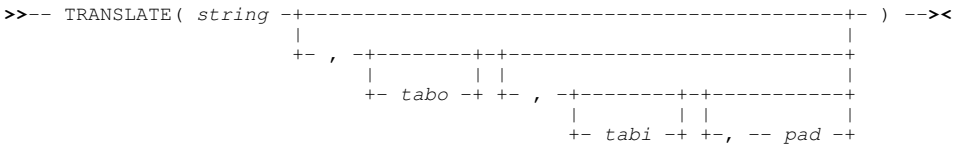
Note that the data type of *string* must be character, *start* and *length* must be a positive integer and *pad* must be a character string of length 1 otherwise ZZSD386E is returned.

Examples:

In the following, field names XText and XLen exist in the focus record type and contain character data `c'The quality of mercy is not strained'` and binary integer value `16` respectively.

```
SUBSTR(XText,29)              =  'strained'
SUBSTR(XText,5,XLen)          =  'quality of mercy'
SUBSTR(XText,29,15,c'x')      =  'strainedxxxxxxx'
SUBSTR(XText,29,15,'x')       =  'strainedXXXXXXX'
```

TRANSLATE()



TRANSLATE() or BIFTRANSLATE() returns the character string *string* with characters at offsets in the input translate table *tabi* translated to characters at equivalent offsets in the output translate table *tabo*. Characters in *string* that are not included in *tabi* are not translated.

If both *tabo* and *tabi* are specified then:

- If *tabo* has a shorter length than *tabi*, then *tabo* is padded on the right to the length of *tabi* using the *pad* character.
- If *tabo* has a longer length than *tabi*, then *tabo* is truncated to the length of *tabi*.

If *tabo* is specified but *tabi* is not, *tabi* defaults to be all 255 single byte characters in the range X'00' to X'FF'.

If *tabi* is specified but *tabo* is not, *tabo* defaults to be a string equal to the length of *tabi* containing only *pad* characters.

If neither *tabo* nor *tabi* are specified, then *string* is simply upper cased as for the UPPER() function call.

If not specified, *pad* defaults to be character blank.

Note that the data type of *string*, *tabo* and *tabi* must be character, and *pad* must be a character string of length 1 otherwise ZZSD386E is returned.

Usage Note:

It is recommended that, when expressed as a literal string, translate tables *tabi* or *tabo* be prefixed with "C" or "c" to avoid unintentional upper casing of lower case translate table alpha characters.

Examples:

In the following, field reference #6 in the focus record type contains character data `c'Ill met by moonlight, proud Titania'`.

```
TRANSLATE(#6)                  =  'ILL MET BY MOONLIGHT, PROUD TITANIA'
TRANSLATE(#6,c'SsXx',c'TtOo')  =  'Ill met by mxxnlighs, prxud Sisania'
TRANSLATE(#6,,c'LlAa',c'#')    =  'I## met by moon#ight, proud Tit#ni#'
TRANSLATE(c'321',c'abc',c'123') =  'cba'
```

TRUNCATED()

```
>>-- TRUNCATED() ------------------------------------------------------------><
```

TRUNCATED() or BIFTRUNCATED() has no arguments and applies to edit of segmented records only. It returns "1" if the focus segment is a primary (base) segment which has been flagged as belonging to a record that has been truncated by an IDENTIFY or a SAVE, FILE or END operation (**=TRNC>**). Otherwise "0" is returned.

Because TRUNCATED() has no arguments, it may be expressed simply as TRUNCATED.

UPPER()

```
>>-- UPPER( --- string --- ) ------------------------------------------------><
```

UPPER() or BIFUPPER() returns the *string* with all lower case alpha characters upper cased and all other characters unchanged.

Note that the data type of *string* must be character otherwise ZZSD386E is returned.

Examples:

In the following, field reference #2 contains character data `c'Mark Williams'`.

```
UPPER('hello')   =  'HELLO'
UPPER(#2)        =  'MARK WILLIAMS'
```

VALUEERROR()

```
>>-- VALUEERROR() ----------------------------------------------------------><
```

VALUEERROR() or BIFVALUEERROR() has no arguments. It returns "1" if the formatted focus record, segment or DB2 table is flagged as including at least one field value which, while valid for the field's data type, is outside the range of valid values defined for the field (**=ERRV>**). Otherwise "0" is returned.

The VALUEERROR flag may be set on when any of the following conditions are true:

1. A field used to define the length of a variable length field has a value greater than the maximum length defined for the variable length field.
2. A field used to define the size of a variable array is outside the range of values specified for the array size.
3. An enum field contains a value not listed as a defined enum value.

Because VALUEERROR() has no arguments, it may be expressed simply as VALUEERROR.

WORD()

```
>>-- WORD( --- string -- , -- #word --- ) ----------------------------------><
```

WORD() or BIFWORD() returns a character string which is equal to word number *#word* in *string*. A word is considered to be a blank delimited token in a character string.

If *#word* is greater than the number of words in *string*, then a null string is returned.

Note that the data type of *string* must be character and *#word* must be a positive integer otherwise ZZSD386E is returned.

Examples:

In the following, field reference #3 in the focus record type contains character data `c'But, soft! what light through yonder window breaks?'`.

```
WORD('  Hello   World ',2)    =  'WORLD'
WORD(#3,2)                    =  'soft!'
WORD(#3,6)                    =  'yonder'
WORD(#3,9)                    =  ''
```

WORDS()

```
>>-- WORDS( --- string --- ) -------------------------------------------------><
```

WORDS() or BIFWORDS() returns the number of words in *string*. A word is considered to be a blank delimited token in a character string.

Note that the data type of *string* must be character otherwise ZZSD386E is returned.

Examples:

In the following, field name Quote_01 exists in the focus record type and contains character data `c'So foul and fair a day I have not seen'`.

```
WORDS(Quote_01)  =  10
WORDS('  ')      =  0
```

XRANGE()

```
>>-- XRANGE( --+-- start --+--+------------+---------------------------------><
               |           |  |            |
               +-----------+  +-, -- end --+
```

XRANGE() or BIFXRANGE() returns a string of single byte character representations between and including *start* and *end*.

If not specified, *start* defaults to X'00' and *end* defaults to X'FF'. If *start* is greater *end*, then the returned string will contain characters from *end* to X'FF' and X'00' to *start*.

Note that the data type of *start* and *end* must be character length 1 otherwise ZZSD386E is returned.

Examples:

```
XRANGE(c'A',c'J')     =  'ABCDEFGHIJ'
XRANGE(X'C1',X'C6')   =  X'C1C2C3C4C5C6'
XRANGE(X'FC',X'03')   =  X'FCFDFEFF010203'
XRANGE(,X'08')        =  X'000102030405060708'
XRANGE(X'FC')         =  X'FCFDFEFF'
```

REXX Macros

As in the CBLe text edit environment, user macros may be written to perform functions within SDE sessions using the REXX procedure language.

The name associated with the Structured Data Environment is CBLSDATA and should be specified on the REXX instruction ADDRESS if commands are to be directed to the SDE environment. However, if a macro is executed from within an SDE window, CBLSDATA is automatically set as the default environment.

The current environment may be identified using the REXX built-in function ADDRESS().

Macro Path

An SDE REXX macro is executed by executing the MACRO command followed by the full fileid of the macro.

Alternatively, if the macro exists in a library within the macro path, simply specify the macro name. Libraries in the macro path are searched in order until a file name that matches the macro name is found.

The macro path used by SDE is the same as that defined for the CBLe text edit environment. i.e. The macro path is a list of directories assigned to the **Edit.MacroPath** variable set via the System or User INI file, or via the CBLe SET MACROPATH command.

For MVS systems, the macro path usually consists of three PDS/PDSE libraries as follow:

1. A user specific macro library.
2. A site-wide macro library common to all users.
3. An installation macro library common to all users and containing useful macros written and owned by CBL and distributed to all SELCOPY Product Suite customers.

   Users should not update or modify macros in this library as they may be subject to change at the discretion of CBL. A detailed description on the use of each of an individual macro is documented within the macro executable file itself. (See CBLe REXX Macros for a list of CBL supplied macros.)

SDE Profile (SDEPROF) Macro

When an SDE window view is opened, the macro path libraries are searched for the first occurrence of the SDE profile macro, **SDEPROF**.

The SDEPROF macro may be used to define the user's SDE environment. Any System or User INI file options that correspond to SDE SET command options, may be overridden for an SDE session by including the SET command in the SDEPROF macro.

A default SDEPROF macro is distributed as part of the SELCOPY Product Suite package and exists within the installation macro library.

SDE Menus and Popup Windows

To assist the user with general SDE editing, some helpful menus and dialog panels exist in addition to the startup CBLe SDE Edit Dialog Window

Library/Directory Member Selection

The SD Edit/Browse Library/Directory Member Selection list window is opened when a PDS(E) library DSN with no member name or HFS directory path with no HFS file name, is specified on the SDE EDIT or BROWSE command.

Select the required entry (position the cursor and hit <Enter>) for Edit/Browse, as specified on the invoking command, or enter select one of the prefix commands.

The window remains open after a selection has been made in order to allow selection of further entries.

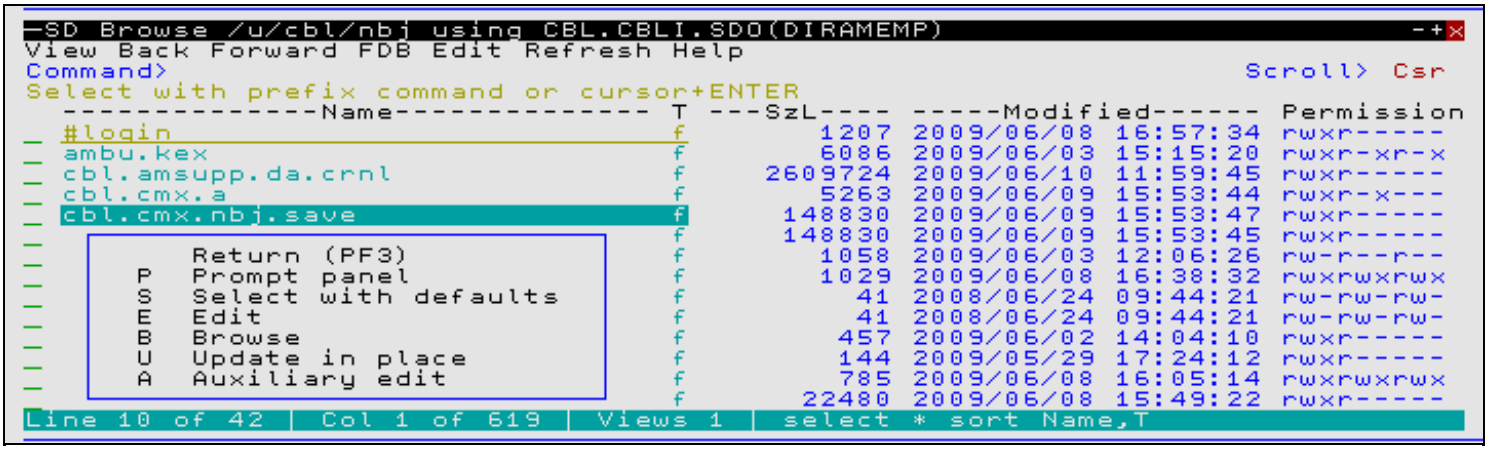

SD Browse /u/cbl/nbj using CBL.CBLI.SDO(DIRAMEMP)
View Back Forward FDB Edit Refresh Help
Command>
Scroll> Csr
Select with prefix command or cursor+ENTER
Name T SzL Modified Permission
#login f 1207 2009/06/08 16:57:34 rwxr-----
ambu.kex f 6086 2009/06/03 15:15:20 rwxr-xr-x
cbl.amsupp.da.crnl f 2609724 2009/06/10 11:59:45 rwxr-----
cbl.cmx.a f 5263 2009/06/09 15:53:44 rwxr-x---
cbl.cmx.nbj.save f 148830 2009/06/09 15:53:47 rwxr-----
148830 2009/06/09 15:53:45 rwxr-----
1058 2009/06/03 12:06:26 rw-r--r--
1029 2009/06/08 16:38:32 rwxrwxrwx
41 2008/06/24 09:44:21 rw-rw-rw-
41 2008/06/24 09:44:21 rw-rw-rw-
457 2009/06/02 14:04:10 rwxr-----
144 2009/05/29 17:24:12 rwxr-----
785 2009/06/08 16:05:14 rwxrwxrwx
22480 2009/06/08 15:49:22 rwxr-----
Return (PF3)
P Prompt panel
S Select with defaults
E Edit
B Browse
U Update in place
A Auxiliary edit
Line 10 of 42
Col 1 of 619
Views 1
select * sort Name,T

Figure 31. SD Edit/Browse Library/Directory Member Selection List Window.

Prefix Line Commands

The following prefix line commands are available:

Command	Description
(blank)	Prefix line command S. (Hit the <Enter> key with the cursor on the line containing the required entry).
A	Edit the entry using Auxiliary Edit.
B	Browse the entry.
E	Edit the entry using Full Edit.
P	Open the CBLe SDE Edit Dialog Window.
S	Select the entry for Edit or Browse as specified by the invoking command.
U	Edit the entry using Update-in-place Edit.
/	Open a drop down menu containing valid prefix command functions for the list window entry. Position the cursor on the required function and hit <Enter> to action the command. Assigned to PF4 by default.
>	Open a new window containing a zoomed vertical display of the entry's fields. Particularly useful for list windows that have a large number of displayed columns. Assigned to PF2 by default.

SDE Browse/Edit Utilities Menu

The SDE Browse/Edit Utilities Menu may be opened from any SDE file edit or browse window view by executing the SDEUTIL macro which is assigned to <PF4> by default.

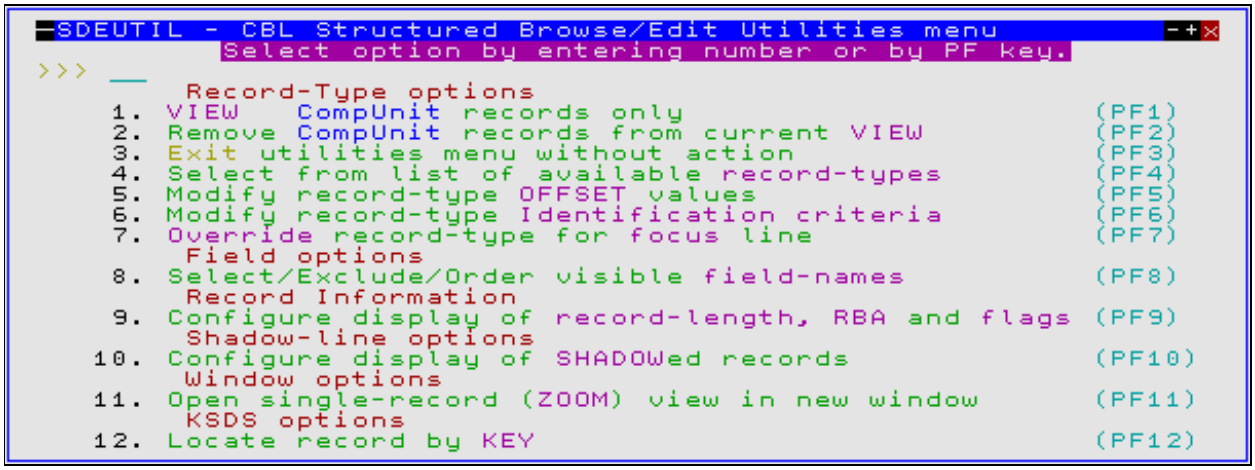

SDEUTIL - CBL Structured Browse/Edit Utilities menu
Select option by entering number or by PF key.
>>>
Record-Type options
1. VIEW CompUnit records only (PF1)
2. Remove CompUnit records from current VIEW (PF2)
3. Exit utilities menu without action (PF3)
4. Select from list of available record-types (PF4)
5. Modify record-type OFFSET values (PF5)
6. Modify record-type Identification criteria (PF6)
7. Override record-type for focus line (PF7)
Field options
8. Select/Exclude/Order visible field-names (PF8)
Record Information
9. Configure display of record-length, RBA and flags (PF9)
Shadow-line options
10. Configure display of SHADOWed records (PF10)
Window options
11. Open single-record (ZOOM) view in new window (PF11)
KSDS options
12. Locate record by KEY (PF12)

Figure 32. SDE Browse/Edit Utilities Window.

This menu provides a user friendly method of executing commonly used SDE commands that alter the display of record data within the SDE window view.

Options are selected by entering the required option number at the prompt or by pressing the appropriate PFKey.

Note that items in the menu are sensitive to the prevailing default record type.

Record-Type Options

`VIEW` *`record_type`* `records only`
: Executes the following SDE VIEW command to make visible only records that are associated with the specified record type (RTO).

```
VIEW  record_type
```

`Add/Remove` *`record_type`* `records to/from current VIEW`
: If the focus line is a shadow line representing a suppressed record group, then this option is "Add" records to the display, otherwise this option is "Remove" records from the display.

The following SDE VIEW command is executed to add to (+) or remove from (-) the display of visible records all records that are associated with the specified record type (RTO).

```
VIEW  (+/-) record_type
```

`Select from list of available record-types`
: Generates a temporary CMX command centre file containing a number of SDE VIEW commands that enables the user to easily add or remove records assigned individual record types from the current display of visible records.

The temporary CMX file is displayed in a CBLe text edit window which allows the user to execute the VIEW commands simply by placing the cursor on the required command (line beginning with "<") and pressing <PF4<.

The CMX file contains the following VIEW commands:

`sd view + *`
: View all records of any record type.

`sd view - *`
: Suppress all records of all record types. Following execution of this command, execute one or more of the "VIEW + *record_type*" commands that follow to display only records of the selected record types.

`sd view Record`
: View only records not assigned a record type. (i.e. NOT SELECTED records.)

`sd view + _record_type`
: Add all records of record type *record_type* to the existing view of records. One of these VIEW commands is generated for every record type (RTO) defined in the structure (SDO).

Execution of any of these VIEW commands will immediately update the display of the records in the current SDE window view. The text edit window containing the CMX file remains open to allow execution of other VIEW commands, until closed (using <PF3<).

`Modify record-type OFFSET values`

Opens the SDEUTOF sub-window which prompts the user to enter a positive or negative numeric offset against any (or all) of the record types (RTOs) defined in the current SDO.

Where an offset value is assigned to a record type, record mapping starts at that offset into (or before) the record data. (See the OFFSET SET/QUERY/EXTRACT option for details.)

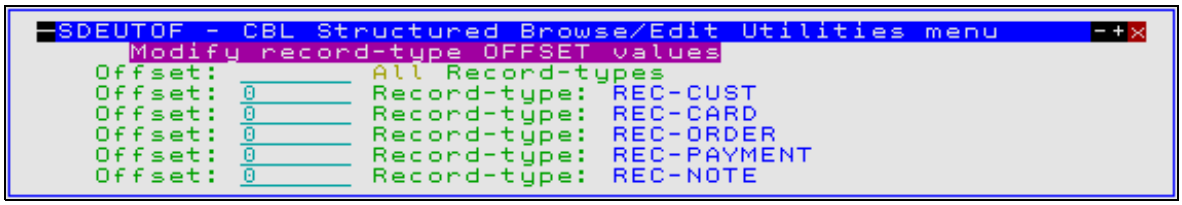

SDEUTOF - CBL Structured Browse/Edit Utilities menu
Modify record-type OFFSET values
Offset: _____ All Record-types
Offset: 0_____ Record-type: REC-CUST
Offset: 0_____ Record-type: REC-CARD
Offset: 0_____ Record-type: REC-ORDER
Offset: 0_____ Record-type: REC-PAYMENT
Offset: 0_____ Record-type: REC-NOTE

Figure 33. SDE Browse/Edit Modify Offsets Window.

<Enter> updates the affected RTO definitions in storage and actions the offset changes immediately. Record type assignment processing is performed for records within the file following a change to any record type's offset value.

<PF3> closes the SDEUTOF window and, if offset changes have occurred, the SDEUTSS sub-window is displayed prompting the user to save the changed structure, overwriting the structure data file on disk, before returning to the utilities menu. If not saved, the user is given another opportunity to save changes to updated structures when the SELCOPY/i session is ended normally.

`Modify record-type Identification criteria`

Opens the SDEUTUS sub-window which prompts the user to enter record type (RTO) selection criteria against any or all record types defined in the SDO.

The Use Always option may set for one record type only. If specified for multiple record types, the first selected will be used. See command USE for descriptions on use of the WHEN *expression*, ALWAYS and NEVER parameters.

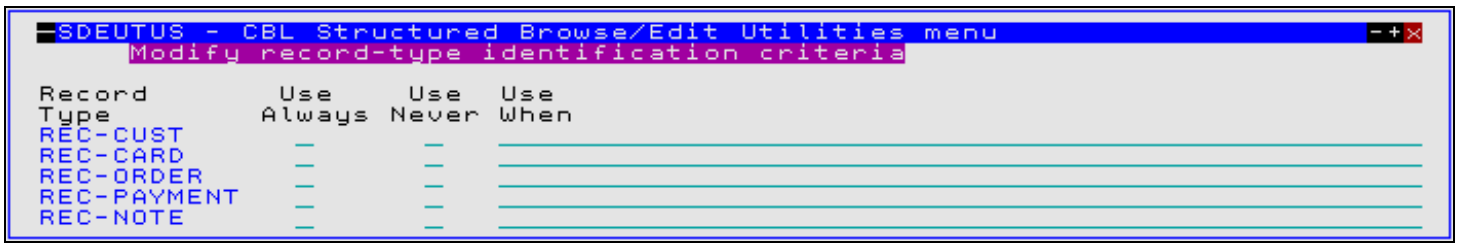


Figure 34. SDE Browse/Edit Record Type Identification Window.

<Enter> updates the affected RTO definitions in storage and immediately actions record type assignment processing for the records within the SDE window view.

<PF3> closes the SDEUTUS window and, if offset changes have occurred, the SDEUTSS sub-window is displayed prompting the user to save the changed structure, overwriting the structure data file on disk, before returning to the utilities menu. If not saved, the user is given another opportunity to save changes to updated structures when the SELCOPY/i session is ended normally.

`Override record-type for focus line`

Opens the SDEUTUF sub-window which prompts the user to select the record type (RTO) to be assigned to the record occupying the focus line. The status of ALWAYS and NEVER flags and any WHEN expressions are displayed in this window for information purposes only.

Only one record type may be selected. If multiple record types are selected, then the last record type in the list to be selected is used. See command USE for descriptions on use of the FOR FOCUS parameter.

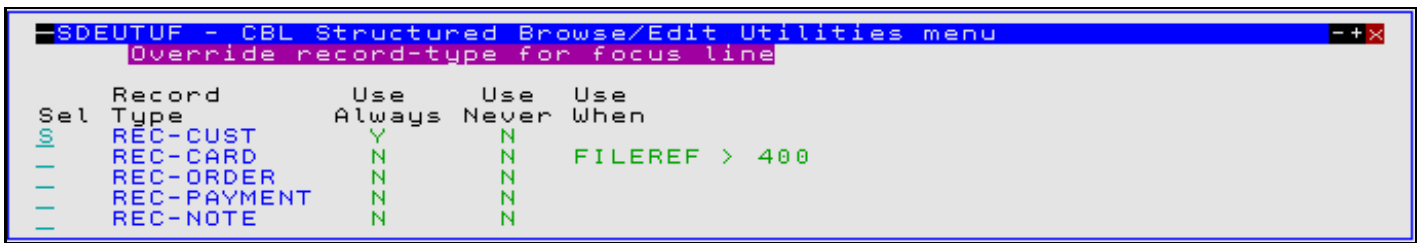


Figure 35. SDE Browse/Edit Focus Line Record Type Identification.

<Enter> applies the record type to the record in the focus line immediatley. <PF3> closes the SDEUTUS window and returns to the utilities menu.

Field Options

`Select/Exclude/Order visible field-names`

Executes SDESEL to generate a temporary CMX command centre file containing an SDE SELECT command that, by default, selects all columns for display in ascending order of field reference number.

The temporary CMX file is displayed in a CBLe text edit window which allows the user to employ standard text editing techniques to re-order and/or exclude the fields from the SELECT command and so select and re-order the display of table row fields.

The SELECT operation may be executed by placing the cursor on the first line of the SELECT command (beginning with "<") and pressing <PF4<.

Record Information

`Configure display of record-length, RBA and flags`

Opens the SDEUTRI sub-window which prompts the user to select the record information to be displayed at the start of all rows in the current SDE window view.

The "Display Record Information" option must be selected before any of the selected record information fields will be displayed.

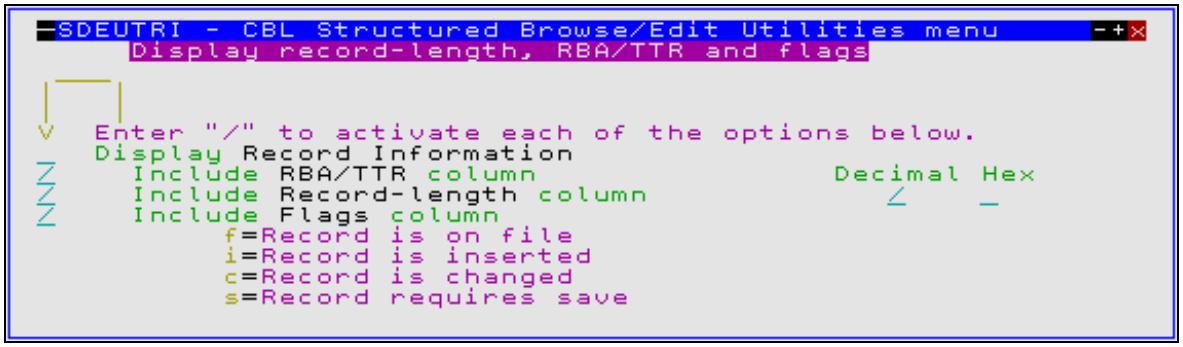

SDEUTRI - CBL Structured Browse/Edit Utilities menu
Display record-length, RBA/TTR and flags
Enter "/" to activate each of the options below.
Display Record Information
Include RBA/TTR column
Decimal Hex
Include Record-length column
Include Flags column
f=Record is on file
i=Record is inserted
c=Record is changed
s=Record requires save

Figure 36. SDE Browse/Edit Record Info Window.

<Enter> actions the information display in the SDE edit view but keeps the SDEUTRI window open. <PF3> closes the window and returns focus to the Utilities Menu.

See the RECINFO SET/QUERY/EXTRACT option for details of the individual record information fields.

Shadow-line Options

`Configure display of SHADOWed records`

Opens the SDEUTSH sub-window which has option fields indicating whether shadow lines are displayed for record groups that are EXCLUDED, NOTSELECTED or SUPPRESSED.

The "Display all SHADOWed records" option must be selected before any of the selected shadow lines will be displayed.

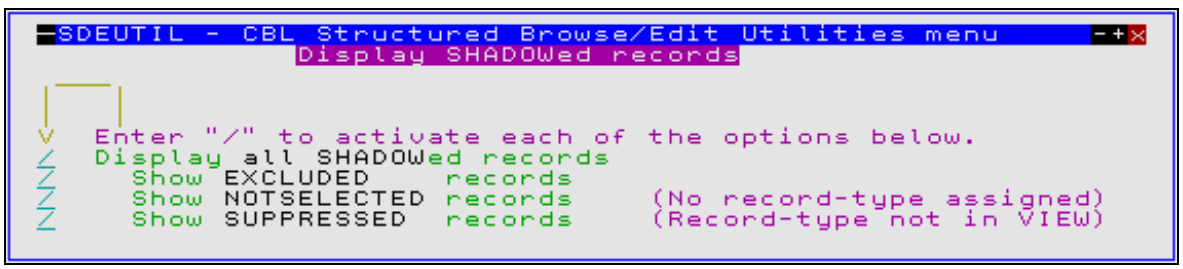

SDEUTIL - CBL Structured Browse/Edit Utilities menu
Display SHADOWed records
Enter "/" to activate each of the options below.
Display all SHADOWed records
Show EXCLUDED records
Show NOTSELECTED records
(No record-type assigned)
Show SUPPRESSED records
(Record-type not in VIEW)

Figure 37. SDE Browse/Edit SHADOW Lines Window.

<Enter> actions the selection for the current SDE window view but keeps the SDEUTSH window open. <PF3> closes the window and returns focus to the Utilities Menu.

See the SHADOW SET/QUERY/EXTRACT option for details of shadow lines.

Window Options

`Open single-record (ZOOM) view in new window`

Executes SDEZOOMW to open another SDE window view containing a single record view of the record occupying the focus line. If the current SDE window view is already in SINGLE format then data in the new view will be a multi-record view of the prevailing format (TABLE, CHAR or HEX.)

See the SDE command ZOOM.

Locate Options

`Locate record by KEY/RBA/RECORD Number`

Locate record by KEY is displayed for KSDS data sets and opens the SDEUTKY sub-window which has a single input field prompting the user for a complete or partial KSDS key identifying the new current record.
The first record with a key which is equal to or greater than this value will become the current record.

Locate record by RBA is displayed for ESDS data sets and opens the SDEUTRB sub-window which has a single input field prompting the user for the relative byte address identifying the new current record.
The first record with an RBA which is equal to or greater than this value will become the current record.

Locate record by RECORD Number is displayed for RRDS and VRDS data sets and opens the SDEUTRR sub-window which has a single input field prompting the user for the record number identifying the new current record.

See the SDE command LOCATE for details on locating records by Key, RBA and Record number.

DB2 Browse/Edit Utilities Menu

The DB2 Browse/Edit Utilities Menu may be opened from any SDE DB2 table edit or browse view by executing the SDEUTILD macro which is assigned to <PF4> by default.

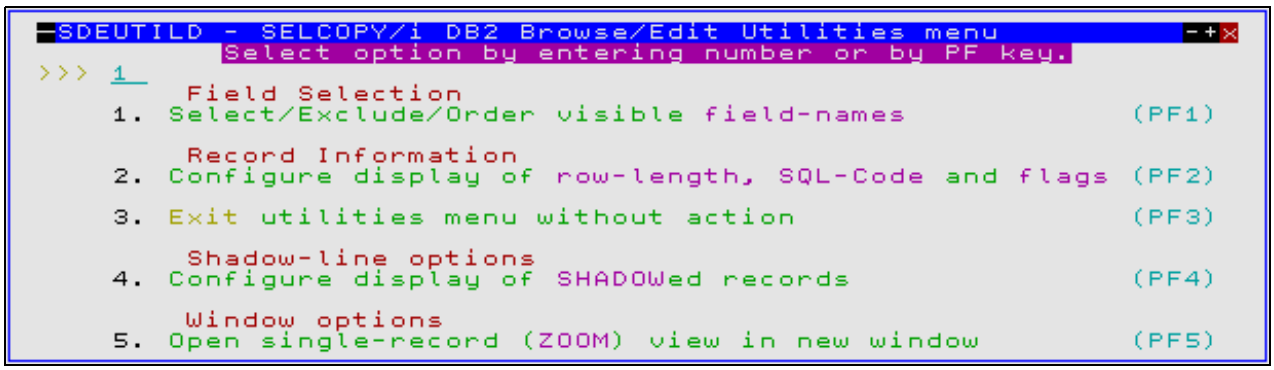

SDEUTILD - SELCOPY/i DB2 Browse/Edit Utilities menu
Select option by entering number or by PF key.
>>> 1
Field Selection
1. Select/Exclude/Order visible field-names (PF1)
Record Information
2. Configure display of row-length, SQL-Code and flags (PF2)
3. Exit utilities menu without action (PF3)
Shadow-line options
4. Configure display of SHADOWed records (PF4)
Window options
5. Open single-record (ZOOM) view in new window (PF5)

Figure 38. DB2 Browse/Edit Utilities Menu.

This menu provides a user friendly method of executing commonly used SDE commands that alter the display of DB2 table data within the SDE window view.

Options are selected by entering the required option number at the prompt or by pressing the appropriate PFKey.

Field Selection

`Select/Exclude/Order visible field-names`

Executes SDESEL to generate a temporary CMX command centre file containing an SDE SELECT command that, by default, selects all columns for display in ascending order of field reference number.

The temporary CMX file is displayed in a CBLe text edit window which allows the user to employ standard text editing techniques to re-order and/or exclude the fields from the SELECT command and so select and re-order the display of table row fields.

The SELECT operation may be executed by placing the cursor on the first line of the SELECT command (beginning with "<") and pressing <PF4<.

Record Information

`Configure display of row-length, SQL-Code and flags`

Opens the SDEUTRI sub-window which prompts the user to select the record information to be displayed at the start of all rows in the current SDE window view.

The "Display Record Information" option must be selected before any of the selected record information fields will be displayed. <Enter> actions the information display in the SDE edit view but keeps the SDEUTRI window open. <PF3> closes the window and returns focus to the Utilities Menu.

See the RECINFO SET/QUERY/EXTRACT option for details of the individual record information fields.

Shadow-line Options

`Configure display of SHADOWed records`

Opens the SDEUTSH sub-window which has a single option field indicating whether shadow lines are displayed for groups of EXCLUDED table rows view.

<Enter> actions the selection for the current SDE window view but keeps the SDEUTSH window open. <PF3> closes the window and returns focus to the Utilities Menu.

See the SHADOW SET/QUERY/EXTRACT option for details of shadow lines.

Window Options

`Open single-record (ZOOM) view in new window`

Executes SDEZOOMW to open another SDE window view containing a single record view of the table row occupying the focus line. If the current SDE window view is already in SINGLE format then data in the new view will be a multi-record TABLE format view.

See the SDE command ZOOM.

Command Line (Primary) Commands

SDE commands may be issued from:

1. The SDE command line.
2. A CBLe text edit window command line using the SDATA command.
3. A CBLe edited text file using the SDATA command and CMDTEXT facility.
4. A CBLe/SDE REXX edit macro.
5. A programmable function key.

Multiple SDE commands may be issued in a single invocation by separating each command with the line end character (set to ";" semi-colon by default.)

Executing SDE Commands from a CBLe Text Edit Window

The CBLe CLI command **SData** may be used to prefix a command stream to be passed to the Structured Data Environment (SDE).

On using this method to execute an SDE command that updates the display of data within an SDE edit view, the current SDE window display gets updated and the focus remains on the current focus CBLe edit view.

EDIT and BROWSE commands will pass focus to the current SDE window.

```
<sdata create structure   CBL.CBLI.STRUCT(COMPSTR)  from cobol CBL.COPYBOOK.COBOL(COMPDEF)
<sd edit  CBL.SDE.EMP   using CBL.CBLI.STRUCT(COMPSTR)
<sd select  Key,InvNumb,DeliveryDate  from Orders   in CBL.CBLI.STRUCT(COMPSTR)
```

Command Reference Syntax Conventions

How to read the Syntax Diagrams

The following rules apply to the syntax diagrams used in this command reference.

1. The diagrams should be read from left to right, from top to bottom, following the path of the line.
   - The >>- symbol indicates the beginning of a statement.
   - The ->< symbol indicates the end of a statement.

2. Required items appear on the horizontal line (the main path).

```
>>--- required_item ---------------------------------><
```

3. Optional items appear below the main path.

```
>>--- required_item ---+----------------------+-----><
                       |                      |
                       +-- optional_item -----+
```

4. If an optional item appears above the main path, then that item has no effect on the execution of the statement and is used only for readability.

```
                       +-- optional_item -----+
                       |                      |
>>--- required_item ---+----------------------+-----><
```

5. If you can choose from two or more items, they appear vertically, in a stack.

6. If you must choose one of the items, one item of the stack appears on the main path.

```
>>--- required_item ---+-- required_choice1 --+-----><
                       |                      |
                       +-- required_choice2 --+
```

7. If choosing one of the items is optional, the entire stack appears below the main path.

```
>>--- required_item ---+----------------------+-----><
                       |                      |
                       +-- optional_choice1 --+
                       |                      |
                       +-- optional_choice2 --+
```

8. If one of the items is the default, it appears above the main path and the remaining choices are shown below.

```
                          +-- default_choice ----+
                          |                      |
>>--- required_item ----+-+----------------------+-----><
                          |                      |
                          +-- optional_choice1 --+
                          |                      |
                          +-- optional_choice2 --+
```

9. An arrow returning to the left, above the main line, indicates an item that can be repeated.

```
                          +----------------------+
                          v                      |
>>--- required_item ----+-+-- repeatable_item ---+-----><
```

10. If the repeat arrow contains a comma, you must separate repeated items with a comma.

```
                          +- , -------------------+
                          v                       |
>>--- required_item ----+-+-- repeatable_item ----+-----><
```

11. A repeat arrow above a stack indicates that you can repeat the items in the stack.

```
                          +----------------------+
                          v                      |
>>--- required_item ----+-+----------------------+-----><
                          |                      |
                          +-- optional_choice1 --+
                          |                      |
                          +-- optional_choice2 --+
```

12. Uppercase characters in keywords indicate the minimum abbreviation allowed for that particular command or parameter and must be spelled exactly as shown.

13. Variables appear in all italic, lowercase letters and represent user-supplied names or values.

14. If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

15. Where a parameter, immediately following a command verb, begins with a non-alpha character, no separating blank is required between the command verb and the parameter. e.g. Add8, CHANGE/abc/xyz/

BOTTOM

Syntax:

```
>>-- BOttom ------------------------------------------------------------------><
```

Description:

Display the last page of data.
Equivalent to the DOWN MAX command.

Parameters:

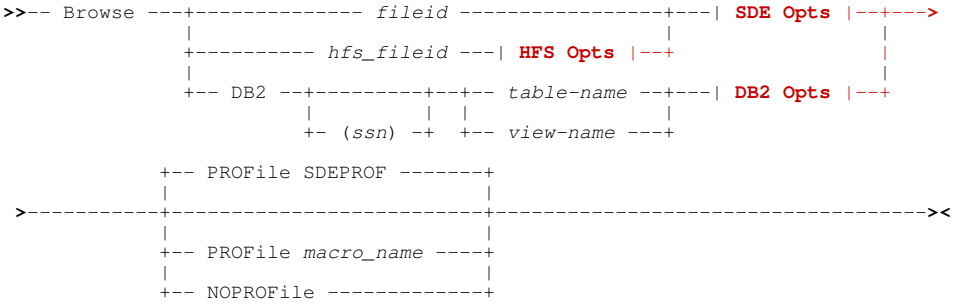
None.

See Also:

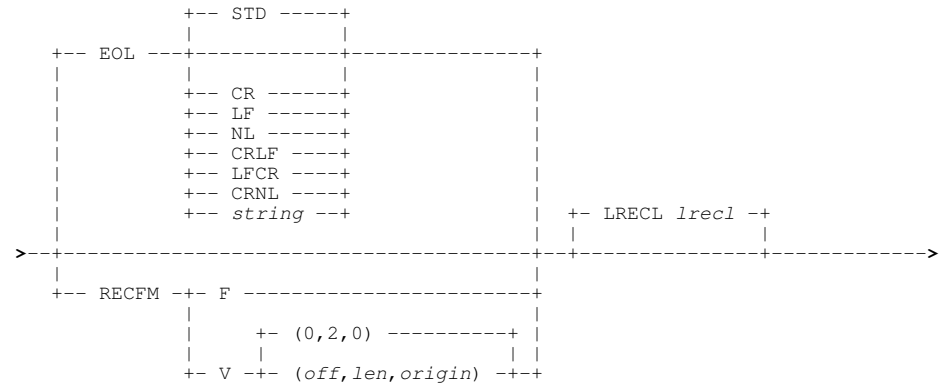
DOWN
TOP

BROWSE

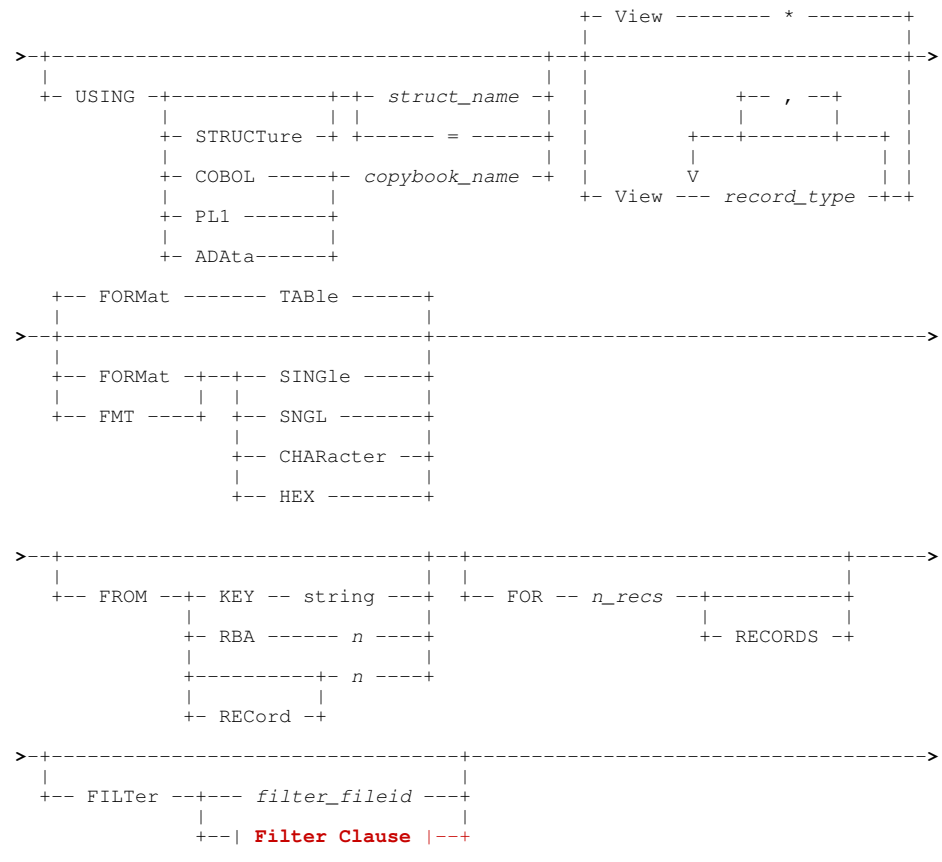
Syntax:



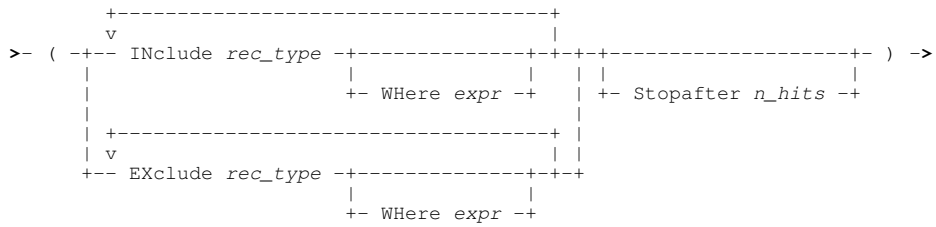
HFS Opts:



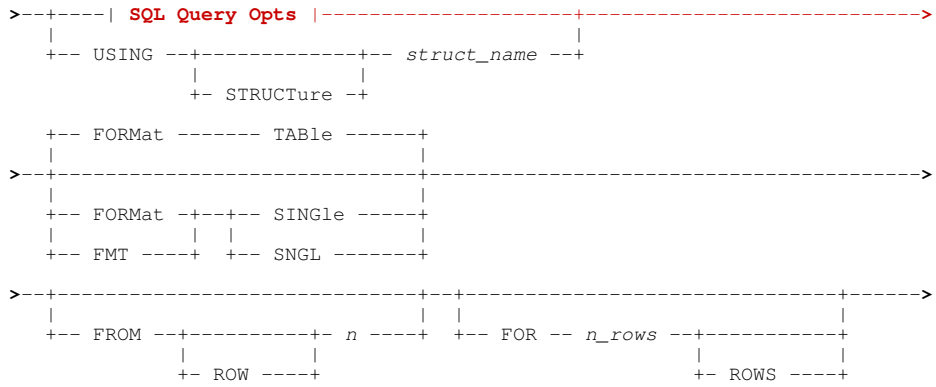
SDE Opts:



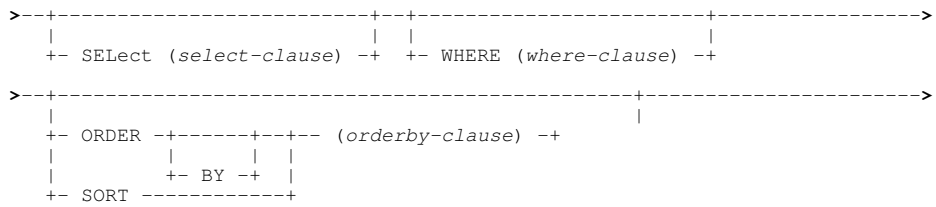
Filter Clause:



DB2 Opts:



SQL Query Opts:



Description:

Browse a structured data set or HFS file, or browse a DB2 results table within SDE. If BROWSE is used, the displayed data cannot be altered in any way.

An SDE edit display window is opened and focus is passed to the new window. Depending on the value on the FORMAT parameter, the SDE display is either a multi record window view or a single record window view for the first record selected.

Structured Data Browse:

Unless FILTER is used, SDE browse will not attempt to load all records of the file into storage, thus allowing quick display of records even in very large data sets. Records are loaded and dropped from storage as required ensuring that available storage is never exceeded. Note that, for both SDE EDIT and BROWSE, FILTER attempts to load all records that match the filter criteria.

If the Structure Definition Object (SDO) specified by the USING STRUCTURE parameter is not already in storage, it is loaded from the appropriate Structure Definition File (SDF). Record Type Objects (RTO) within the specified SDO are then used to map the records in the data set.

For every record in the file, each RTO is checked until one is found whose criteria is matched by the record characteristics. This RTO is then used to map the record. If the record does not match any RTO criteria, then the first RTO in the SDO is used by default. The RTO automatically selected to map a record may be changed by the user via the USE WHEN command or the SDE Edit/Browse Utility Window.

By default, all fields within records that have an associated RTO are displayed as printable character, numeric data fields having first been converted to decimal.

If no SDO is specified, then FORMAT CHARACTER is applied to each record. i.e. the data is displayed as a single, variable (RECFM=V) or fixed (RECFM=F) length character field with field name "UnMapped". No data conversion is performed.

If the VIEW parameter is specified with a non-generic argument, the DRECTYPE setting is initialised as the first *record_type* parameter specified on the VIEW parameter. Otherwise, the DRECTYPE setting is initialised as the record type of the first visible record in the file.

Record data that does not meet the specifications of its field definition is considered to be invalid and is represented by asterisks.

DB2 Table Browse:

DB2 table browse is supported for in-storage edit/browse only. Therefore, if a results table is too large to be loaded into available storage a storage error occurs. The user must then restrict the number of rows to be loaded using further row selection criteria (i.e. FROM, FOR, SELECT, WHERE parameters).

On starting a DB2 table browse, a temporary SDO is created reflecting field characteristics obtained from the SQL Descriptor Area return by the prepared SQL SELECT execution. Furthermore, if the user does not already have an open connection to the required DB2 sub-system that is not for an SDE DB2 table edit view, then a new DB2 connection is made.

Parameters:

fileid
The fileid of the data set containing records to be browsed.
fileid may be the DSN of a sequential or VSAM data set or the DSN and member name of a PDS/PDSE member.

hfs_fileid
A complete HFS (or zFS) file path containing data to be browsed.
hfs_fileid is identified as an HFS path if *fileid* is not a valid DSN for a sequential, VSAM or PDS/PDSE dataset. e.g. DSN begins with ".", contains invalid special characters (e.g. "/") or contains qualifiers of length greater than 8.

HFS Opts
The following HFS options may be specified to determine the handling of data within the HFS file.

EOL=STD|NL|CR|LF|CRLF|LFCR|CRNL|*string*
Sets the EOLIN (input end-of-line) delimiter value used to determine the end of each record for non-RECFM F/V input. EOLIN delimiters are not included in the browsed record data or record length. EOL parameter elements are as follow:

STD	-	Any standard line-end.
NL	X'15'	New Line.
CR	X'0D'	Carriage Return.
LF	X'0A'	Line Feed.
string	-	A 2-byte user specified character or hex string.

STD is default so that the BROWSE operation scans the input data for any of the standard EOL combinations (not *string*), stopping when one is found. This EOL combination is used as EOLIN for the file.

Where EOL processing is specifically requested for BROWSE, if EOL characters are **not** found within the first lrecl bytes of data, then the file is displayed as RECFM F with records of length 80.

RECFM F | V (*off*,*len*,*origin*)
Specifies that the data is to be treated as containing Fixed or Variable length format records.

RECFM F indicates that all records are of a fixed length as defined by the LRECL argument.

RECFM V allows the user to specify the location of the record length fields within the data as follows:

off	Offset of the record length field from the start of the record.
len	Length of the record length field.
origin	The start of the record data at which the record length is applied.

Default is (0,2,0) which describes standard RECFM V organisation data sets.

LRECL *lrecl*
Specifies the maximum record length of input records.

Records terminated by an EOL sequence will wrap onto the next line of data if the record length exceeds *lrecl*. Where a record has wrapped, the prefix area contains the "==EOL>" flag.

For RECFM F data, *lrecl* is the fixed length of the records in the edit view. The data within the last record is padded with blanks up to the *lrecl* length if necessary.

If the record length field of a RECFM V record exceeds the *lrecl* value, then an error is returned.

RECFM V and EOL delimited records have default *lrecl* of 32752, wheras RECFM F records have default *lrecl* of 80.

SDE Opts
The following SDE options are applicable to browse of structured data found in HFS files or z/OS data sets only. See DB2 Opts for DB2 table browse options.

`USING (STRUCTURE) struct_name`
Identifies the SDO to be applied to the data records. *struct_name* is the SDF fileid assigned to the SDO in a CREATE STRUCTURE command.

"=" (equals) may be specified instead of *struct_name* to indicate the name of the structure being used in the current SDE window.

`USING COBOL | PL1 | ADATA copybook_name`
Identifies the full fileid of *copybook_name*, a COBOL/PL1 copy book or COBOL/PL1 ADATA output file to be used to format the data records.

Using *copybook_name* syntax has a processing overhead in that a CREATE STRUCTURE operation is performed to generate a temporary SDO. This overhead is greater if COBOL or PL1 is specified since a compile of the copybook is performed to first obtain the ADATA file output. Therefore, it is recommended that a non-temporary SDO is generated using CREATE STRUCTURE and that this is used for future formatting of the data (on EDIT, BROWSE, FSU, etc.)

`VIEW record_type`
`VIEW *`
Identifies one or more record types for which records will be displayed in the initial SDE view. Records that are associated with other record types are not visible within the display but are displayed as shadow lines instead. * (asterisk) indicates all record types including the internal record type "UnMapped" (or "UnMappedSeg") which is used to map data records or record segments that have no associated RTO in the SDO.

Records within the display may be later suppressed or made visible using the VIEW command.

Default is VIEW *.

`FORMAT`
`FMT`
Specifies the format in which record data will be displayed in the initial SDE view.

`SINGLE` `SNGL`	Single record format.
`TABLE`	Table format. (Default)
`CHARACTER`	Multi record view with all records or record segments mapped as single, variable length character fields with field name "UnMapped" or "UnMappedSeg". No data conversion is performed.
`HEX`	Same as CHARACTER with the addition that the data is also displayed in Hex below the character display. Note that the Hex display occupies an additional 2 lines of data.

The data display format may be later altered using the FORMAT and/or ZOOM commands.

`FROM`
Records are to be displayed beginning at a specific location within the data set. The first record at the specified location becomes record 1 within the SDE window view of the data set. Records that occur before this location are not included within the browse session.

If FILTER is specified, then record filtering occurs only on records selected using the FROM and FOR parameters.

Required record location is identified via one of the following:

`KEY string`	For KSDS data sets only, locate the record with a key string equal to *string*. If not found, the record with the next key string greater than *string* is used.
`RBA n`	For KSDS and ESDS data sets only, locate the record starting at the relative byte address specified by *n*. The RBA must point at the start of the record otherwise a VSAM point error will occur. RBA address for each record is displayed as part of the record information columns within an SDE window view. Display of these columns is controlled using the SDE SET RECINFO CLI command.
`< RECORD > n`	For any data set organisation, locate the record number specified by *n*. Specification of parameter keyword RECORD is optional.

Default is to display records within the SDE window view starting at the first record in the data set.

`FOR n_recs < RECORDS >`
Specifies the number of data set records to be included within the SDE window browse session. Specification of parameter keyword RECORDS is optional.

If FILTER is specified, then record filtering occurs only on records selected using the FROM and FOR parameters.

Default is to include all records.

`FILTER filter_fileid | Filter Clause`
FILTER specifies additional record filtering criteria to further reduce the display of records for browse. All record filters are applied only to records that have been selected using the FROM and/or FOR parameters, otherwise it applies to all records in the structured data file.

Enough available storage must exist to load all records selected by the filter otherwise a storage error occurs and the browse is terminated.

FILTER parameters are specified via a filter clause which may be supplied as part of the BROWSE command or referenced via *filter_fileid*, a separate sequential data set, PDS/PDSE member or HFS file. *filter_fileid* must contain the keyword FILTER followed by a valid filter clause.

The filter is applied only once during initial load of the data. A record will not be excluded from the display of browsed records if its record type or data is changed in a way that no longer satisfies the filter criteria.

Filter Clause

A filter clause must be specified in "( )" (parentheses) and may contain comment data enclosed by "/*" and "*/". If filter clause is specified via *filter_fileid*, then comment data may also occur before and after the filter clause.

The following options are supported by the filter clause.

`INCLUDE record_type`
Include only records that are assigned the specified record type *record_type* by SDE browse. This parameter may be specified repeatedly to include a number of record types or to perform alternative WHERE *expr* filters for the same record type. If INCLUDE is specified, then all record types that are not referenced by an INCLUDE parameter will be excluded by default.

record_type "Record" (with field name "UnMapped") may be used to perform a filter on the unformatted record data whether or not a structure (USING *struct_name*) has been specified. In this way, a filter may test **all** records regardless of their assigned record type.

INCLUDE and EXCLUDE parameters are mutually exclusive.

`EXCLUDE record_type`
Exclude only records that are assigned the specified record type *record_type* by SDE browse. This parameter may be specified repeatedly to exclude a number of record types or to perform alternative WHERE *expr* filters for the same record type. If EXCLUDE is specified, then all record types that are not referenced by an EXCLUDE parameter will be included by default.

record_type "Record" (with field name "UnMapped") may be used to perform a filter on the unformatted record data whether or not a structure (USING *struct_name*) has been specified. In this way, a filter may test **all** records regardless of their assigned record type.

INCLUDE and EXCLUDE parameters are mutually exclusive.

`WHERE expr`
WHERE applies further filter conditions to records assigned to the record type specified by the last INCLUDE *record_type* or EXCLUDE *record_type* parameter processed.

expr is a valid SDE expression which supports function calls, *record_type* field names and references, sub-expressions, arithmetic, relational and logical operators. The result of the WHERE expression must be numeric and is treated as being Boolean in nature with a zero value indicating a "false" condition and any non-zero value indicating a "true" condition.

The WHERE expression is applied to each record assigned the record type *record_type* and, if the result is "true", the record is selected for include or exclude as indicated by the prevailing INCLUDE or EXCLUDE filter. If multiple INCLUDE/EXCLUDE *record_type* WHERE expressions exist for the same record type, then a logical OR is implied for all the expressions relating to that record type.

`STOPAFTER n_hits`
When the number of records selected by the INCLUDE or EXCLUDE filter reaches the value specified by STOPAFTER *n_hits*, then no further filter testing occurs.

If an INCLUDE filter, then all remaining untested records are excluded. If an EXCLUDE filter, then all remaining untested records are included.

`DB2 < (ssn) >`
Indicates that browse is for a DB2 base or results table. *ssn* is optional and identifies the local DB2 sub-system name to which a connection will be made.

Before a connection can be made to the DB2 sub-system, the SELCOPY/i DB2 plan must have been bound to that sub-system.

Default for *ssn* is the users default DB2 sub-system name as set by the DB2 Primary Options menu.

`table_name`
A DB2 base table name, specified as *table_owner.table_name*, containing rows of data to be browsed.

`view_name`
A DB2 table view, specified as *view_owner.view_name*, which references an SQL query used to generate a results table containing the rows of data to be browsed.

DB2 Opts

The following DB2 options are applicable to browse of DB2 table data only. See SDE Opts for structured data browse options.

`SELECT (select-clause)`
Specifies a DB2 SQL SELECT clause to be included in the prepared SQL select statement. See "*DB2 SQL Reference*" for syntax of the *select-clause*.

`WHERE (where-clause)`
Specifies a DB2 SQL WHERE clause to be included in the prepared SQL select statement. See "*DB2 SQL Reference*" for syntax of the *where-clause*.

`SORT|ORDER BY (order-by-clause)`
Specifies a DB2 SQL ORDER BY clause to be included in the prepared SQL select statement. See "*DB2 SQL Reference*" for syntax of the *order-by-clause*.

`USING (STRUCTURE) struct_name`
Identifies the SDO to be applied to the DB2 data rows. *struct_name* is the SDF fileid assigned to the SDO in a CREATE STRUCTURE command.

If the USING STRUCTURE option is specified then SELECT, WHERE and ORDER BY options are ignored. The SDO may contain DB2 SQL query SELECT, WHERE and ORDER BY sub-clause options which are used to select and display DB2 table rows.

`FROM < ROW > n`
Equivalent to FROM RECORD *n*, FROM ROW *n* specifies that rows are to be displayed beginning at row number *n* in the DB2 results table returned by the SQL query. The first row displayed becomes row 1 within the SDE window browse view. Rows that occur before this row number are not included within the browse session. Default is row 1.

`FOR n_rows < ROWS >`
Equivalent to FOR *n_recs* RECORDS, FOR *n_rows* ROWS specifies the maximum number of rows to be displayed from the DB2 results table returned by the SQL query. Rows that fall outside the range of rows specified by FROM *n* FOR *n_rows* are not included within the browse session.
Default is to display all selected rows.

`PROFILE macro_name`
Specifies the REXX SDE edit macro to be executed as the profile when the data set is browsed.

macro_name must exist in a library within the SDE macro path.

The PROFILE option only alters the profile used for the file currently being browsed. It does not define the profile macro to be used for subsequent SDE edit or browse.

The default is to execute a macro with member name SDEPROF if it exists.

`NOPROFILE`
Suppresses use of a profile macro when browsing the file.

The NOPROFILE option only suppresses use of a profile for the file currently being browsed. It does not suppress use of a profile macro for subsequent SDE edit or browse.

Examples:

```
<sd browse CBL.SDE.MOD.ZJ2202   using CBL.CBLI.SDO(PRODL)  view ARCX1,PRODX
```
Browse records from data set "CBL.SDE.MOD.ZJ2202" within the SDE.
Initially display all records of type ARCX1 and PRODX where the record type objects (RTO) are found in the structure definition object (SDO) referenced by CBL.CBLI.SDO(PRODL).

```
<sd browse  db2(DB9G)  CBL.APIFUNC  where(FUNCMOD#REF = 0)
```
Browse filtered rows of DB2 table CBL.APIFUNC in subsystem DB9G.

See Also:

EDIT
CREATE STRUCTURE

CANCEL

Syntax:

```
>>--- CANcel ------------------------------------------------------------><
```

Description:

Close **all** SDE window views for data displayed in the current SDE view. Any unsaved changes to the data will be discarded.

If a non-temporary structure (SDO) is used to map the record data and changes have been made to that structure during the course of the edit session (e.g. USE WHEN), then the user will be prompted to save the changed structure to its structured data file (SDF). See command, SAVESTRUCTURE.

Parameters:

CANCEL has no parameters.

See Also:

END
FILE
QQUIT

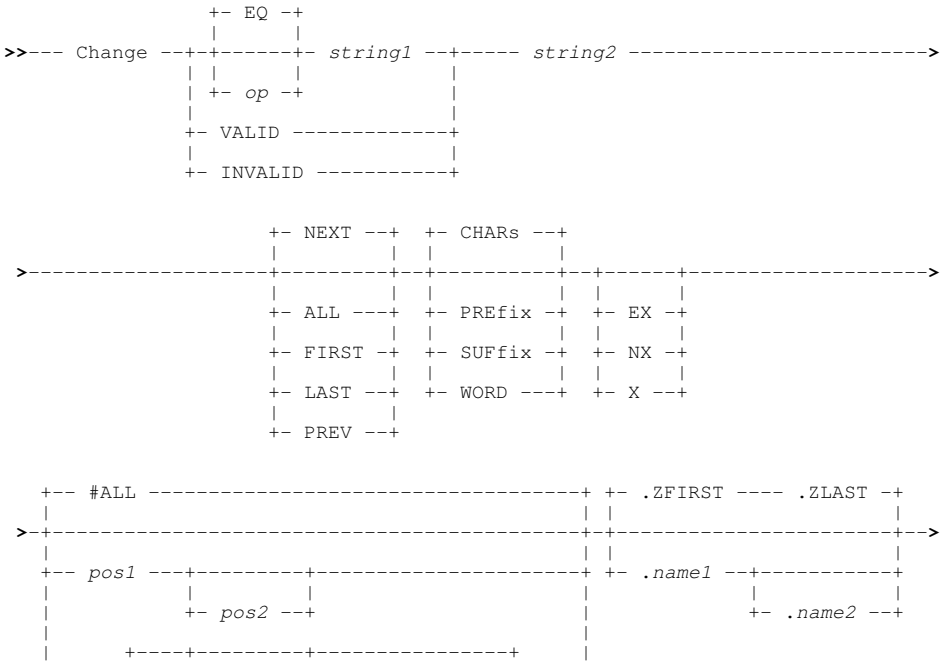
CBLI

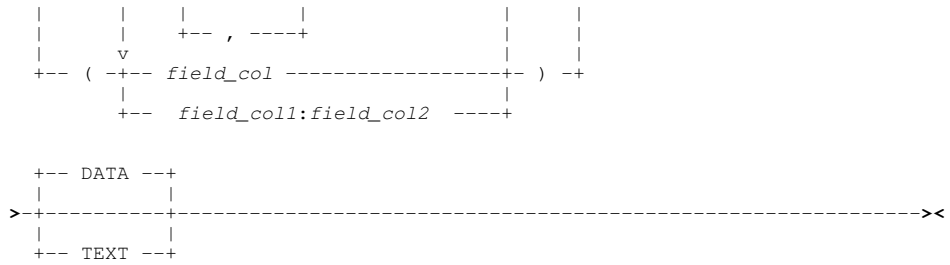
Description:

SDE CLI command, CBLI, performs the same operation as the CBLe CLI command CBLI. See CBLI in CBLe Text Edit documentation.

CHANGE

Syntax:


+- EQ -+
>>--- Change --+-+------+- string1 --+----- string2 ------------------------->
| | | |
| +- op -+ |
| |
+- VALID ------------+
| |
+- INVALID ----------+
+- NEXT --+ +- CHARs --+
>--------------------+---------+--+----------+--+------+--------------------->
| | | | | | |
+- ALL ---+ +- PREfix -+ +- EX -+
| | | | | |
+- FIRST -+ +- SUFfix -+ +- NX -+
| | | | | |
+- LAST --+ +- WORD ---+ +- X --+
| |
+- PREV --+
+-- #ALL -------------------------------------+ +- .ZFIRST ---- .ZLAST -+
| | | | |
>-+---+-+----------------------+-->
| | | | |
+-- pos1 ---+---------+---------------------+ +- .name1 --+-----------+
| | | | | | |
| +- pos2 --+ | | +- .name2 --+
| | |
| +----+---------+----------------+ |



Description:

Search data records in the current edit or browse view for the specified character string or numeric value (*string1*) and replace it with *string2*. Only data records that are of the default record type (visible and EXCLUDED records) are included in the search. Records groups that are of a different record type or are NOTSELECTED or SUPPRESSED, are excluded from the search.

If the specified occurrence (all, first, last, next or previous) of the search string or numeric value (*string1*) is found within a record, then:

1. If the record is EXCLUDED, it is made visible.
2. The cursor is positioned at the beginning of the string or numeric field.
3. If necessary, scrolling occurs to display the found data.
4. If the replace string (*string2*) satisfies the data type, precision and scale requirements for the field, then *string2* replaces *string2* in the field.

All occurrences of *string1* are highlighted in the text. (Enter the RESET FIND command to turn off the highlighting.)

Use of CHANGE with no parameters is equivalent to RCHANGE (assigned to function key **PF6** by default) and repeats the last change command executed, including all its specified parameters.

To find and optionally change the next occurrence of *string1*, execute a combination of RFIND (assigned to PF5 by default) followed by RCHANGE.

If *string1* and *string2* are character strings of unequal length, then the following will occur:

- If the length of *string1* is greater than the length of *string2*, then words to the right of the replaced string will be shifted left.

  If parameter TEXT is specified and more than one blank exists before a word to the right of the replaced string, then additional blanks are inserted to maintain that word's position in the record.

- If the length of *string1* is less than the length of *string2*, then words to the right of the replaced string will be shifted right. Note, however, that CHANGE will not increase the length of formatted data beyond its defined maximum field length.

  If parameter TEXT is specified, multiple, consecutive blanks are absorbed to leave at least one blank between each word. Only if no blanks are eligible to be absorbed will text to the right of the replaced string be shifted right.

If a structure has been applied to the edited records and a CHANGE operation alters the length of a record or updates a field referenced by a USE *record_type* WHEN condition, then the record type (RTO) associated with the changed record is re-evaluated and the contents of the SDE edit view updated accordingly.

When the duration of a CHANGE or RCHANGE execution exceeds 1 second, a progress window is opened displaying the number of records processed so far. This window also provides the user with the oportunity to interrupt the operation using the Attn key.

The FORMAT of the SDE display affects the execution of CHANGE.

FORMAT CHAR and HEX:

A character compare for the supplied search string is performed against entire, unformatted data records.

field_col and #ALL parameters are not applicable to these display formats and are ignored.

FORMAT SINGLE and TABLE:

The search string is compared against **individual fields** that have been selected for display in the expanded data record. (See SELECT)

Fields are searched from left to right in the order that they appear in the display. This is true regardless of the order in which field columns are specified on the CHANGE command, or the order in which fields are encountered within the expanded record.

If a field column is specified on the CHANGE command that is not part of the display (e.g. an expanded group field or a field that has been removed from display by a SELECT command), then an error message is returned.

For character data type fields, a string compare is performed. For numeric data type fields (binary, packed decimal, floating point, zoned, etc.), then the following will occur:

1. If *pos1*, *pos2* positional parameters are **not** specified, the search string is interpreted as a signed numeric value and an arithmetic compare for equality is performed against the field's formatted numeric value.

   The length and data type of the numeric field, and the number of digits in the search value are not significant. e.g.

   ```
   CHANGE  67   23
   ```

   Finds numeric fields with value "67" (e.g. "0067", "67.00", "0.0670E+03") and character fields containing the string "67" (e.g. "167 Baker Street") and replaces with the value "23" using the appropriate data type

   If the search string is non-numeric, then numeric data type fields are not searched. Therefore, to bypass searching numeric data type fields, explicitly set the search string to be character or hex using 'string', C'string' or X'string' formats respectively. e.g.

   ```
   CHANGE   '67'     '23'
   CHANGE  C'67'    C'23'
   CHANGE  X'F6F7'  X'F2F3'
   ```

   Finds only character fields containing the string "67" and replaces with character string "23".

2. If *pos1*, *pos2* positional parameters are specified, the search string is interpreted as being a character string and a string compare is performed on the character representation of numeric data.

   Although the range of positions may span a number of fields, the search is still performed against individual fields within the range. i.e. a match for the search string will not be found if the data spans a field boundary.

   A match for the search string on just part of the character representation of a numeric field is considered to be a successful hit and the whole numeric field is highlighted. e.g.

   ```
   CHANGE   476   850    21  100
   ```

   Finds character and numeric fields where character representation of the data contains the string "476" and replaces it with "850". (e.g. A zoned decimal field with value "14760" would become "85060")

Where a character string or numeric value is found in a numeric field, the entire field is highlighted and the cursor positioned at the start of the field.

If replacing a string in a character field would exceed the defined maximum length of the field then no update will take place.

Parameters:

op

A relational operator used in the compare operation which determines the relationship that the data must have with the search string in order for it to be identified as a successful match.

Valid values for *op* are as follow:

Operator	Description
EQ	Data must be equal to *string1*. (Default)
NE	Data must be not equal to *string1*.
GT	Data must be greater than *string1*.
GE	Data must be greater than or equal to *string1*.
LT	Data must be less than *string1*.
LE	Data must be less than or equal to *string1*.

If a character string compare is performed, the EBCDIC values assigned to characters in the search and data strings determine the relationship (equal to, greater than or less than) between the two strings.

string1

The CHANGE search string. The seach string may be any of the following:

- ◊ An unquoted numeric value. The search string is treated as a numeric value when a numeric field is searched in a SINGLE or TABLE format view. In all other cases a numeric search string is treated as a character string.
- ◊ An unquoted character string containing no commas or blanks. The search for the character string will be case-insensitive so that uppercase and lowercase characters are treated as being the same.
- ◊ A character string enclosed in single (') or double (") quotation marks. The search string may contain embedded commas and blanks and the character string will be case-insensitive. A string enclosed in quotes may still be interpreted as a numeric value.

  Two adjacent quotation mark characters that are embedded in a search string which is enclosed by the same quotation mark characters, will be treated as a single occurrence of the character. e.g.

  ```
  CHANGE  'Jim O''Brien'  'James O''Brien'
  ```

  Find the character string "Jim O'Brien" and replaces it with "James O'Brien".

◊ A character string enclosed in single (') or double (") quotation marks with the prefix C. This is equivalent to specifying a quoted search string but that the string search will be case-sensitive. (e.g. C'Book')

◊ A hexadecimal string enclosed in single (') or double (") quotation marks with the prefix X.

◊ A picture string enclosed in single (') or double (") quotation marks with the prefix P.

Picture strings use special characters to represent a generic group of characters as described below. Any character in a picture string that is not one of these special characters is untranslated.

String	Description
P'='	Any character.
P'¬'	Any non-blank character.
P'.'	Any non-displayable character.
P'#'	Any numeric character, 0-9.
P'-'	Any non-numeric character.
P'@'	Any uppercase or lowercase alpha character.
P'<'	Any lowercase alpha character.
P'>'	Any uppercase alpha character.
P'$'	Any non-alphanumeric special character.

VALID

Intended for use with formatted records, VALID will search fields for valid data. i.e. data that satisfies the field's assigned data type.

INVALID

Intended for use with formatted records, INVALID will search fields for invalid data. i.e. data that does not satisfy the field's assigned data type.

string2

The CHANGE replace string used to replace *string1*. The replace string may be any of the following:

◊ An unquoted numeric value. The replace string is treated as a numeric value when it replaces a value in a numeric field in a SINGLE or TABLE format view. In all other cases a numeric replace string is treated as a character string.

◊ An unquoted character string containing no commas or blanks. String2 may be null ('').

◊ A character string enclosed in single (') or double (") quotation marks that may contain embedded commas and blanks. String2 may be null (C'').

Two adjacent quotation mark characters that are embedded in a replace string which is enclosed by the same quotation mark characters, will be treated as a single occurrence of the character. (See example under *string1*.)

◊ A character string enclosed in single (') or double (") quotation marks with the prefix C. This is equivalent to specifying a quoted search string but that the string search will be case-sensitive. (e.g. C'Book')

◊ A hexadecimal string enclosed in single (') or double (") quotation marks with the prefix X.

◊ A picture string enclosed in single (') or double (") quotation marks with the prefix P.

Where a picture string is used as *string2* of a CHANGE command, then it must be the same length as the *string1* and may only contain the following special characters. Any character in a picture string that is not one of the special characters supported by picture strings, is untranslated.

String	Description
P'='	Same as the corresponding character in the search string.
P'<'	Change the corresponding character in the search string to lowercase.
P'>'	Change the corresponding character in the search string to uppercase.

ALL

Where field conditions are satisfied, change all occurrences of *string1* to *string2*. A message is displayed providing the number of occurrences of *string1* that have been changed to *string2*. If NX is not specified, all excluded records that contain an occurrence of the *string1* are made visible whether or not *string1* is replaced.

FIRST

Search forwards from the top of the file data (i.e. the first position of the first data record) to find the first occurrence of *string1* and attempt to replace it with *string2*.

LAST

Search backwards from the bottom of the file data (i.e. the last position of the last data record) to find the last occurrence of *string1* and attempt to replace it with *string2*.

NEXT

Search forwards from the current cursor location to find the next occurrence of *string1* and attempt to replace it with *string2*. If the cursor is not within the window's data display area, the search begins at the first position of the first visible or excluded record within the display area that is of the default record type.

PREV
: Search backwards from the current cursor location to find the previous occurrence of of *string1* and attempt to replace it with *string2*. If the cursor is not within the window's data display area, the backwards search begins at the first position of the first visible or excluded record within the display area that is of the default record type.

CHARS
: For non-numeric search strings only, CHARS indicates that a successful match occurs if *string1* is found anywhere within the data being searched.

PREFIX
: For non-numeric search strings only, PREFIX indicates that a successful match only occurs if *string1* is found at the start of a word within the data being searched. i.e. the matched text must precede an alphanumeric character and either be preceded by a non-alphanumeric character or be at the start of a line or field.

SUFFIX
: For non-numeric search strings only, SUFFIX indicates that a successful match only occurs if *string1* is found at the end of a word within the data being searched. i.e. the matched text must be preceded by an alphanumeric character and either precede a non-alphanumeric character or be at the end of a line or field.

WORD
: For non-numeric search strings only, WORD indicates that a successful match only occurs if *string1* is found to be a complete word within the data being searched. i.e. the matched text must either be preceded by a non-alphanumeric character or be at the start of a line or field, and either precede a non-alphanumeric character or be at the end of a line or field.

EX
X
: Search EXCLUDED data records only. By default, both visible and EXCLUDED records are searched. CHANGE does not search records that are NOTSELECTED or SUPPRESSED.

NX
: Search only visible data records (i.e. not EXCLUDED). By default, both visible and EXCLUDED records are searched. CHANGE does not search records that are NOTSELECTED or SUPPRESSED.

pos1
: The first position of a range of positions within the data record to be searched.

  For SINGLE and TABLE formats, this is a position in the expanded record. Only those fields, or parts of fields, that fall within the position range will be searched. Fields will be searched in the order that they occur within the display area.

  pos1 may be a positive or negative integer value (not zero) and must be a value that is less than or equal to the maximum length of the data records or, for formatted record data, the length of the expanded record.

  A negative value represents a position in the record relative to the end of the record. Therefore, where position 1 references the 1st character in the record, position -1 references the last character.

  For all display formats, *string1* is treated as being non-numeric and the search is actioned on the character representation of the record data.

pos2
: The last position of a range of positions within the data record to be searched.

  Like *pos1*, *pos2* may be a positive or negative integer value (not zero). If *pos1* references a position within the record data which is higher than that referenced by *pos2*, then the *pos1* and *pos2* values are swapped.

  If *pos2* is greater than the maximum length of the data records or, for formatted record data, greater than the length of the expanded record, then *pos2* is set equal to the maximum (or expanded) record length.

  Default is *pos1* plus the length of the search string minus 1.

#ALL
: Search all field columns in the current TABLE or SINGLE format display.

field_col
: An individual field column to be searched within a TABLE or SINGLE format display.

  The field column may be identified by its reference number (e.g. #6) or its name (e.g. JobID). If a field name is used, enclosing parentheses are **mandatory** Field names are automatically converted to a field reference and Referencing the same field column more than once will not cause an error.

  If the field is an array, then all elements of the array are searched. To search an individual element of an array, the element occurrence should be specified in parentheses as a subscript to the field reference/name. e.g. #13(3) for the 3rd element of a single dimension array. An entry must exist for each dimension of the array. e.g. RoomSize(6,2,4) - a three dimensional array.

  Specification of multiple field columns must either be enclosed in parentheses and/or separated by commas. The field columns may be specified in any order, however, the data is always searched from the first column in the display to the last.

  Field column search specifications may be a combination of individual field columns and columns ranges. e.g. (JobID #6 Tax_Reference #12 #15:#20).

A search is only performed on field columns that are selected for display. Therefore any field column specified on the CHANGE command that has not been selected for display, will be ignored.

field_col1:*field_col2*
The first and last fields of a range of field columns to be searched within a TABLE or SINGLE format display.

field_col1 and *field_col2* must be separated by ":" (colon) and if *field_col1* occurs after *field_col2* in the data record, then the values are swapped. *field_col1* and *field_col2* have the same specifications as *field_col*.

Specification of multiple field column ranges must either be enclosed in parentheses and/or separated by commas. Field column search specifications may be a combination of individual field columns and field columns ranges. e.g. (FirstName:LastName, #2, Salary:Bonus, EmpNo, #10:#15). If field names are used, enclosing parentheses are **mandatory**.

.name1
A label name identifying the first record of a range of data records to be searched. The preceding "." (dot) is mandatory. Default is .ZFIRST.

.name2
A label name identifying the last record of a range of data records to be searched. The preceding "." (dot) is mandatory. If *.name2* occurs before *.name1* in the display, then the order is reversed.
If CHANGE PREV is executed and *.name1* is specified, the default is .ZFIRST. Otherwise the default is .ZLAST.

DATA
DATA (as opposed to TEXT) indicates that records are to be treated as data so that CHANGE performs no special treatment of any multiple, consecutive blanks that follow the replaced string.
DATA is default.

TEXT
TEXT (as opposed to DATA) indicates that records are to be treated as formatted text so that CHANGE operaties identically to the ISPF edit style CHANGE command. i.e blanks following the replaced string may be absorbed or added in order to maintain text positions.

Examples:

```
change  22.3  303.25
```
Search all fields in the display belonging to the default record type for the next occurrence of the search string/value 22.3. Numeric fields are tested for numeric value 22.3, character fields are tested for character string "22.3" anywhere within the field. Where the field width and, for numeric fields, the precision and scale definitions are satisfied by the replacement string/value 303.25, text update will takes place.

```
change  all  'ing' 's'  suffix  ex  (DsName:ProcLib, JobStep)
```
Search selected character fields in the display that belong to excluded records of the default record type, for all occurrences of the word suffix string "ing" and replace the text with "s".

See Also:

EXCLUDE
FIND
RCHANGE
RFIND
SELECT
Find and Replace Data

CHAR

Syntax:

```
>>-- CHAR ------------------------------------------------------------------><
```

Description:

Display records or record segments in multi-record, unformatted character view by temporarily disabling the record-type (RTO) assigned to each record or record segment.

Compare with FORMAT CHAR which displays the records/segments in multi-record, unformatted view but does not disable the assigned record-type.

Note that unformatted records have record type "UnMapped", unformatted segments of a segmented record have record type "UnMappedSeg".

Parameters:

CHAR has no parameters.

See Also:

FORMAT
MAP (FMT)
UNFMT
VFMT
ZOOM

CLIPBOARD

Description:

SDE CLI command, CLIPBOARD, performs the same operation as the CBLe CLI command CLIPBOARD. See CLIPBOARD in CBLe Text Edit documentation.

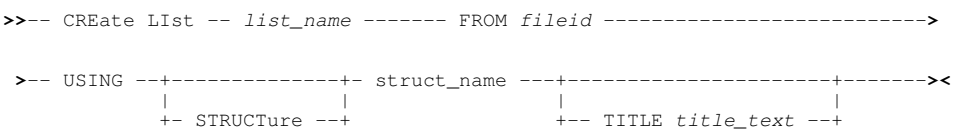
CMSG

Description:

SDE CLI command, CMSG, performs the same operation as the CBLe CLI command CMSG. See CMSG in CBLe Text Edit documentation.

CREATE LIST

Syntax:



Description:

Create an in storage list from records within a structured data set, that may then be displayed in a SELCOPY/i List window via the DISPLAY LIST command.

Only records of one record type may be accurately displayed within a list window. By default, the first record type object (RTO) within the SDO is used to map **all** the records within the file when generating the list.

A list column entry is generated for every field name defined by the RTO. Therefore, data within named fields that are part of a named group of fields will be repeated in different columns.

The List window displaying the list data will support most of the standard list window features as follow:

- Field Descriptor Block (FDB)
- Edit View
- Selecting, Sorting and Filtering
- Sorting with the Cursor

On exiting the List window display, the in-storage list is destroyed and storage is freed.

Parameters:

list_name
The name to be assigned to the in-storage list data for reference in a DISPLAY LIST command.

`FROM fileid`
Identifies the fileid of the data set containing the structured records.
fileid may be the DSN of a sequential or VSAM data set or the DSN of a PDS/PDSE member.

`USING (STRUCTURE) struct_name`
Identifies the SDO to be applied to the data records. *struct_name* is the SDF fileid assigned to the SDO in a CREATE STRUCTURE command.

`TITLE title_text`
Specifies the text to be displayed in the List window's title bar.

Examples:

```
<sd create  list  AMLIST   from CBL.AMSUPP.DA   using CBL.CBLI.SDO(AMMAP)   title "Contact Address Details"
```
Generate in-storage list AMLIST.

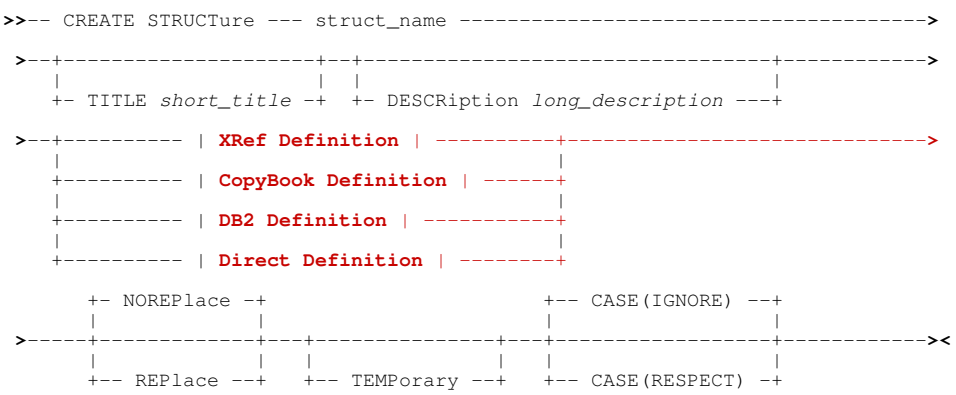
```
<sd display list  AMLIST
```
Open a list window for the in-storage list AMLIST.

See Also:

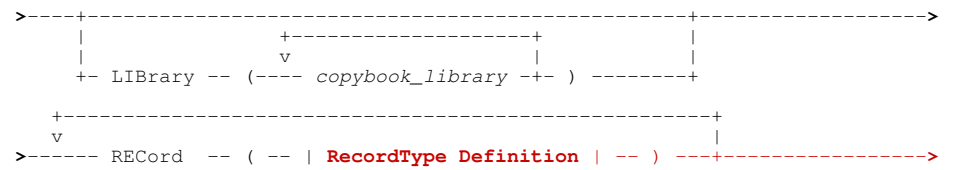
DISPLAY LIST
CREATE STRUCTURE

CREATE STRUCTURE

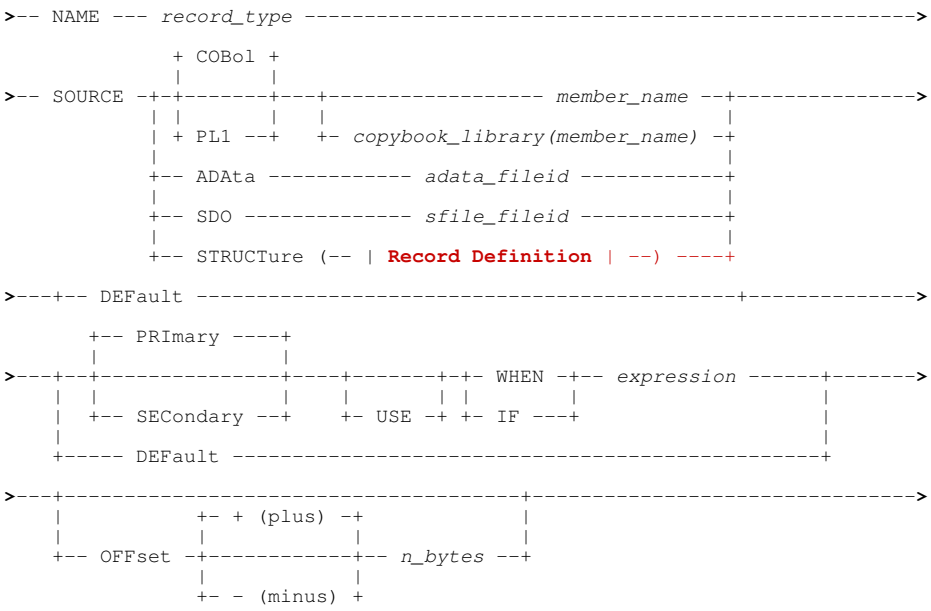
Syntax:



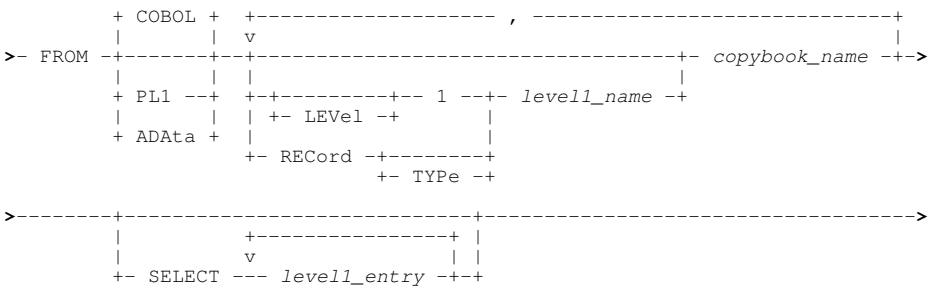
XRef Definition:



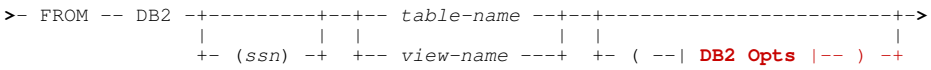
RecordType Definition:



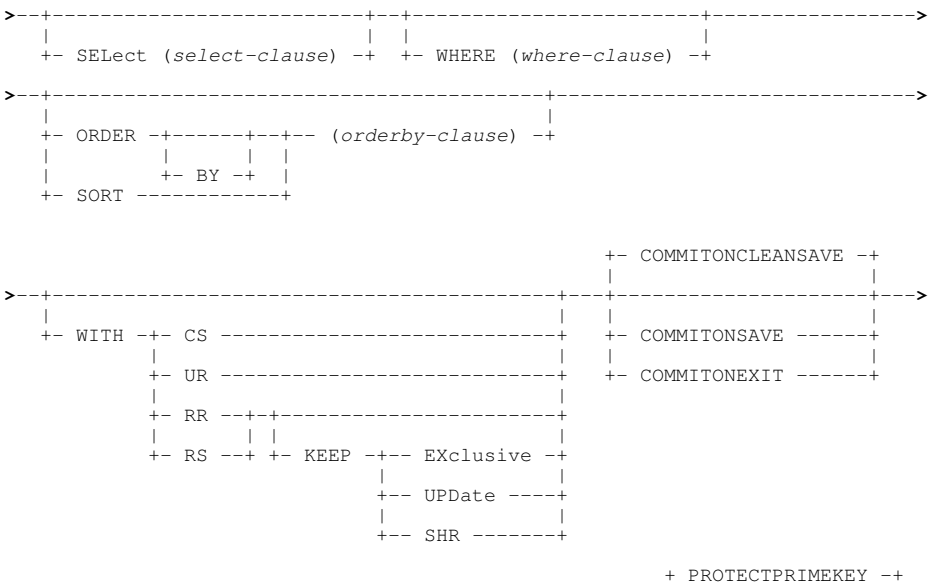
CopyBook Definition:

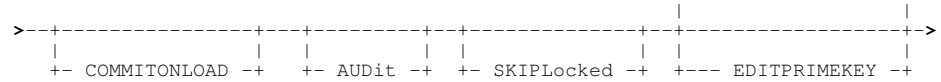


DB2 Definition:

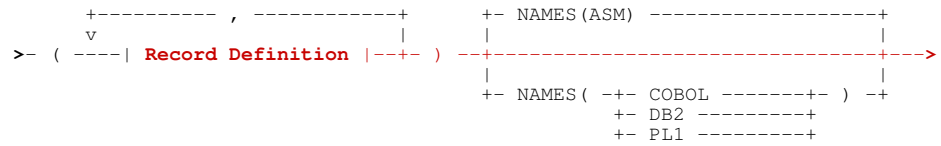


DB2 Opts:

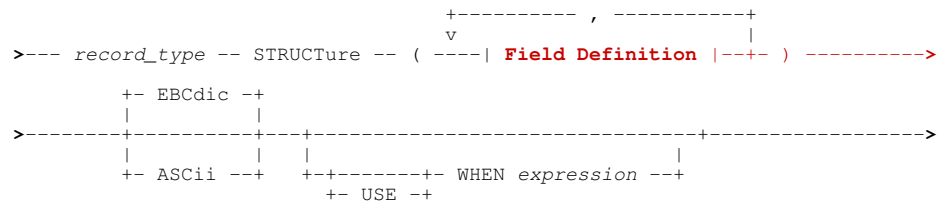




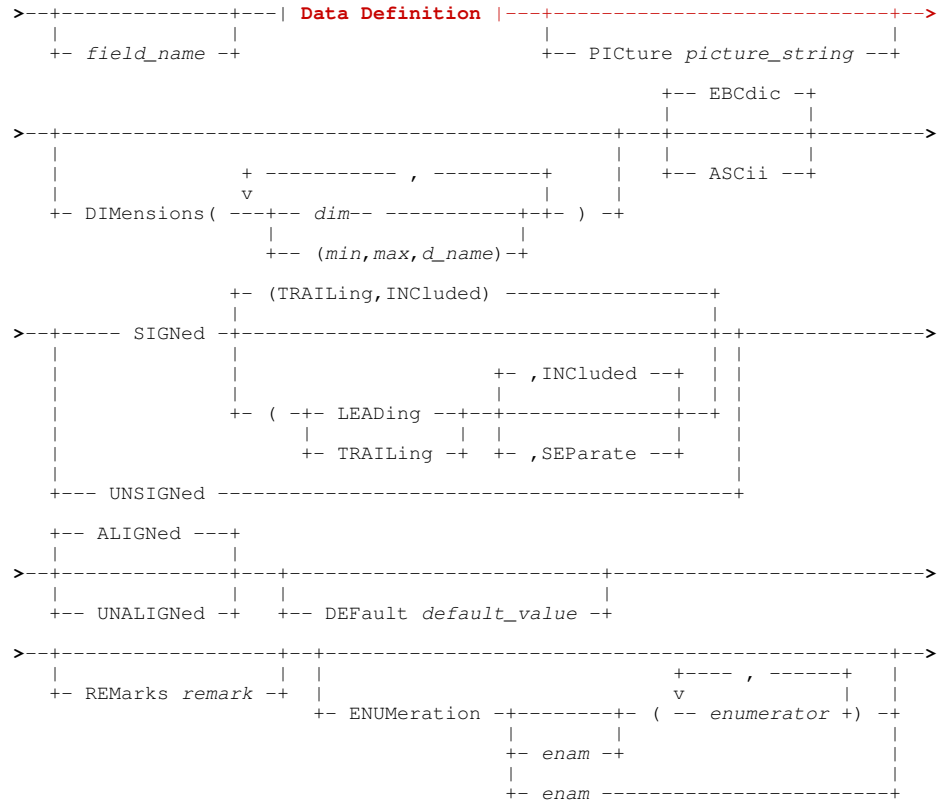
Direct Definition:



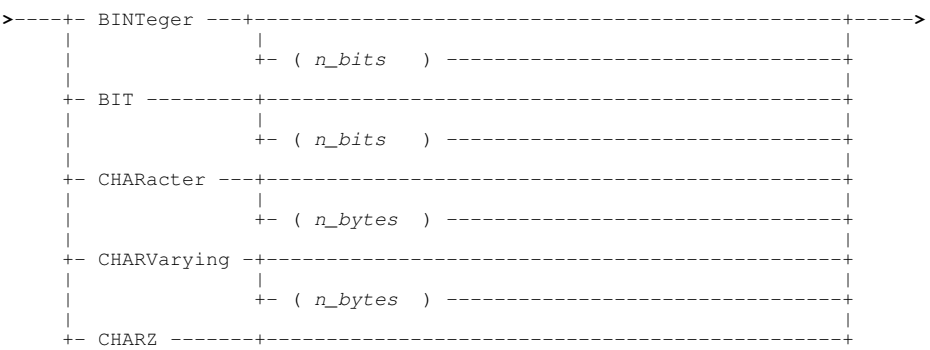
Record Definition:

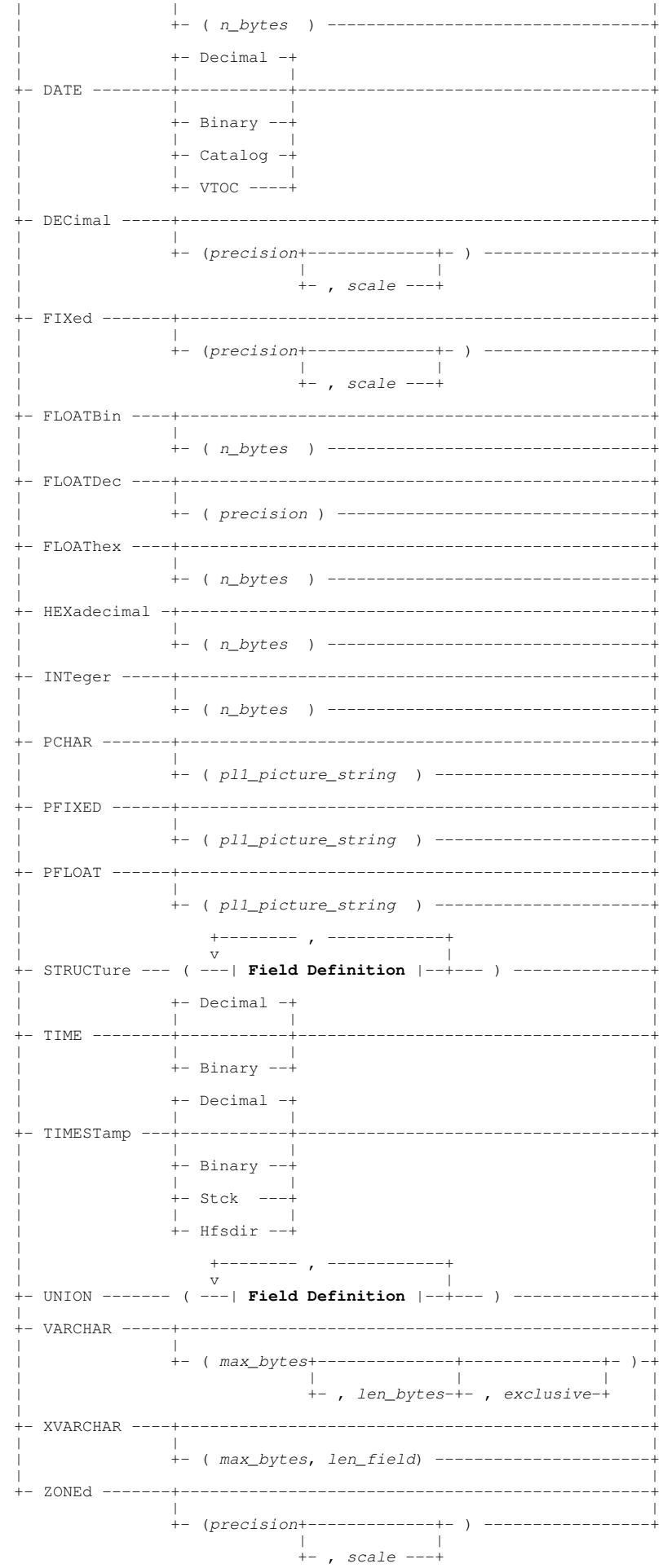


Field Definition:



Data Definition:




(n_bytes)
Decimal
DATE
Binary
Catalog
VTOC
DECimal
(precision , scale)
FIXed
(precision , scale)
FLOATBin
(n_bytes)
FLOATDec
(precision)
FLOAThex
(n_bytes)
HEXadecimal
(n_bytes)
INTeger
(n_bytes)
PCHAR
(pl1_picture_string)
PFIXED
(pl1_picture_string)
PFLOAT
(pl1_picture_string)
STRUCTure
Field Definition
TIME
Decimal
Binary
TIMESTamp
Decimal
Binary
Stck
Hfsdir
UNION
Field Definition
VARCHAR
(max_bytes , len_bytes , exclusive)
XVARCHAR
(max_bytes, len_field)
ZONEd
(precision , scale)

Description:

Create a Structure Definition Object (SDO) to be used within SDE, and, if TEMPORARY is not specified, automatically save it to a Structure Definition File (SDF). An SDO provides the information required to interpret indiviudal logical fields within a file's data records.

Usually a CREATE STRUCTURE command will refer to a file structure already defined within an existing COBOL or PL/1 copybook, but direct definition syntax is also supported (see *record_definition* below).

Parameters:

struct_name
The name assigned to the structure being created, *struct_name* is the SDF fileid assigned to the SDO which will subsequently be referenced via the *USING* parameter of *EDIT* and *BROWSE* commands.

Unless parameter TEMPORARY is specified, *struct_name* must be the DSN of a new or existing data set or PDS(E) member to which the file structure definition is saved.

TITLE *short_title*
Specifies a title to be assigned to the structure.

The structure's title is displayed as part of the structure (SDO) list and may be displayed or updated via the SET/QUERY/EXTRACT TITLE option.

The *short_title* text string should be no longer than 64 characters and may be enclosed in quotation marks (") or apostrophes (').

DESCRIPTION *long_description*
Specifies a description to be assigned to the structure.

The structure's description and may be displayed or updated via the SET/QUERY/EXTRACT DESCRIPTION option.

The *long_description* text string should be no longer than 512 characters and may be enclosed in quotation marks (") or apostrophes (').

REPLACE
NOREPLACE
Either overwrite or give an error if *struct_name* already exists.
Default is NOREPLACE.

TEMPORARY
Create the SDO in storage without automatically saving it to an SDF data set. A temporary SDO cannot be saved.

Where TEMPORARY is used, *struct_name* is simply a named reference for the SDO and does not need to be a valid DSN. As for standard SDOs, when a temporary SDO is created, it will remain in storage until it is dropped using the DROP or DDROP command or the CBLe Edit session is ended.
Default is to save the SDO to an SDF allocated to DSN *struct_name*.

CASE(IGNORE)
CASE(RESPECT)
Respect or ignore the case setting of any variable names used in the structure definition. Default is **CASE(IGNORE)**.

XRef Definition
XRef Definition syntax is an alternative, more flexible method of generating a SELCOPY/i structure (SDO) than using CopyBook Definition or Direct Definition syntax.

Features of XRef Definition which are not supported by these other methods are:

1. Define segment record type structures for mapping the individual segments (primary and secondary) of segmented records.
2. Define a record type based on any COBOL group or PL1 major/minor structure within a COBOL/PL1 copybook or ADATA file.
3. In a single CREATE STRUCTURE execution, define multiple record types from a mixture of COBOL and/or PL1 copybooks, ADATA files, existing SELCOPY/i structures (SDOs) and/or in-line record type definition syntax.
4. Specification of a default record type.
5. Assignment of USE WHEN selection criteria to each non-default primary and secondary record type definition generated from COBOL/PL1 source. (This is mandatory.)

LIBRARY (*copybook_library ...*)
Specifies a library search chain for COBOL or PL1 copybook lookup. If a library containing a required source copy book is in the search chain, then *copybook_library* need not be specified following the SOURCE COBOL/PL1 parameter.

One or more blank separated *copybook_library* PDS/PDSE library data set names are specified in "( )" (parentheses).

`RECORD ( RecordType Definition )`
A separate RECORD specification is required for each record type to be defined within the structure.

RecordType Definition
RecordType Definition incorporates all parameters supported by each RECORD parameter specification in XRef Definition.

`NAME record_type`
Identifies the name of the record type to be defined in the SDO.

Unless SOURCE references SELCOPY/i SDE record definition syntax via the STRUCTURE sub-parameter, *record_type* must match the name of a group (major or minor structure) field in the source (copybook, ADATA, SDO) object.

`PRIMARY`
Identifies the record type as being a non-default primary (base) segment.

A primary segment maps data at the start (first segment) of a segmented record or, for non-segmented records, maps all data in a record. Primary segment definitions are never selected to map data beginning at any position other than position 1 of a segmented record.

Specification of USE WHEN criteria is mandatory in order to identify the conditions by which this record type is allocated to a record/base segment.

`SECONDARY`
Applicable to segmented records only, SECONDARY identifies the record type as being a secondary segment.

A secondary segment maps data that immediately follows any primary segment or any other secondary segment defined within the structure. Secondary segment definitions are never selected to map data beginning at position 1 of a segmented record.

Specification of USE WHEN criteria is mandatory in order to identify the conditions by which this record type is allocated to a secondary segment.

If no secondary segment record type criteria is satisfied by the record data, then the remainder of that record is displayed using the default secondary segment "UnmappedSeg". This maps the remaining segmented record data as a character data field of length equal to the remaining record data length.

`DEFAULT`
Identifies the record type as being the default primary (base) segment.

DEFAULT may be assigned tp only one record type definition in the SDO. Furthermore, USE WHEN criteria must not be specified for default record type definitions.

The default record type is assigned to a record (or primary segment) when it does not satisfy the USE WHEN selection criteria of any of the defined primary record type definitions.

`<USE> WHEN|IF expression`
Specifies the selection criteria that must be satisfied by the record (or record segment) data, in order for it to be assigned the associated record type mapping.

The USE WHEN *expression* is an SDE expression supporting field value references, built-in functions, sub-expressions

The USE WHEN *expression* is a valid SDE expression which supports function calls, record type field names and references, sub-expressions, arithmetic, relational and logical operators.

The result of the USE WHEN expression must be numeric and is treated as being Boolean in nature with a zero value indicating a "false" condition and any non-zero value indicating a "true" condition.

`SOURCE`
The SOURCE parameter identifies the input file (data set or PDS/PDSE library member) and the type of input to be used to generate the record type definition.

If COBOL or PL1, then the appropriate compiler will be invoked to compile the source file and, if the compiler return code is within the range of acceptable values (See options MAXCOBOLRC and MAXPL1RC), use the resulting ADATA file to build the record type definition (RTO). If the source is an existing COBOL or PL1 ADATA output file, then the RTO is generated directly from the ADATA source (no compilation is required.)

SDO source may be used to copy an RTO definition from an existing SDO structure, whereas STRUCTURE indicates that the RTO is to be constructed using in-line record definition syntax.

`COBOL|PL1 member_name|copybook_library(member_name)`
Identify a COBOL or PL1 copy book containing the group field definition (major or minor structure) whose name matches *record_type* specified by the NAME parameter.

copybook_library(member_name) specifies the fully qualified PDS/PDSE library DSN and member name. If a library search chain is provided via the LIBRARY parameter, then *member_name* may be specified without the library DSN. In this case the first instance of *member_name* found within the search chain, will be used.

`ADATA adata_fileid`
Identify a COBOL or PL1 ADATA output file, *adata_fileid*, which contains the group field definition (major or minor structure) whose name matches *record_type* specified by the NAME parameter.

`SDO sfile_fileid`
Identify a previously defined SELCOPY/i SDE structure (SDO) file, *sfile_fileid*, which contains the record type whose name matches *record_type* specified by the NAME parameter.

`STRUCTURE (Record Definition)`
Specifies the CREATE STRUCTURE Record Definition syntax used to define a record type of name *record_type* specified by the NAME parameter.

`OFFSET <+|-> n_bytes`
Specifies a positive or negative offset from the start of the record (or segment) at which the record type will start mapping data.

Use of an offset is documented in detail under SET/QUERY/EXTRACT option OFFSET.

CopyBook Definition
Defines the one or more external copy book(s) or an ADATA compile output file to be used to generate the SDO. Multiple copy books are concatenated to produce a single copy book and the then individual copy book record mappings (RTOs) identified as follow:

1. Each "01" level entry will generate a new record type (RTO) mapping.
2. For **COBOL** copy books only, if no "01" level entry exists within the entire concatenation of copy books and the next available lowest level entry is **redefined** by all entries of the same level within the copy book, then these entries are treated as "01" levels and a new record type (RTO) mapping generated for each.
3. For **COBOL** copy books only, if either of the following are true, then an "01" level entry with name "SELCOPY-01" or *level1_name* (see parameter LEVEL 1 or RECORD TYPE below) is generated by SDE at the start of the copy book concatenation.
   - The first data declaration statement is not an "01" level entry but at least one "01" level entry exists within the concatenation of copy books.
   - No "01" level entry exists within the entire concatenation of copy books and the next available lowest level entry is **not redefined** by all entries of the same level within the copy book. In this case, the created structure will contain a single record type (RTO) mapping.

`FROM COBOL|PL1|ADATA`
Identifies the input as a copy book with programming language COBOL or PL/1. Alternatively identifies input as a COBOL or PL/1 ADATA file.

COBOL and PL/1 copy books that have been compiled using the following compiler options, generate ADATA output.

ADATA	COBOL compiler option.
XINFO(SYN,SYM)	PL/1 compiler option.

Using an already generated ADATA output reduces the inevitable overhead that occurs if SELCOPY/i itself has to re-compile a copy book in order to create the SDO.

Default is COBOL.

`(LEVEL) 1 level1_name`
`RECORD (TYPE) level1_name`
Insert an appropriate COBOL or PL/1 level 01 record (Record Type), with the field name specified by *level1_name*, immediately before the specified copy book data.
This is intended for use where no level 01 record is defined within the copy book.

`copybook_name`
The complete fileid (DSN and member name) of a COBOL or PL/1 copy book, or ADATA file.

`SELECT level1_entry`
Selects the individual "01" level entry names from within the concatenation of specified copy books for which record type mappings (RTOs) are to be generated. All data declaration entries associated with "01" level entries that are not selected, will be ignored.
Default is to generate an RTO for each "01" level entry in the copy book concatenation.

DB2 Definition
DB2 definition syntax provides the method of defining an SDO containing a single record type structure (RTO) based on columns returned by a prepared SQL query statement. CREATE STRUCTURE uses information stored in the returned

SQLDA to determine the data type, etc. for each field definition in the RTO. To do this, a BIND of the SELCOPY/i DB2 plan must have been performed for any DB2 sub-system containing tables involved in the CREATE STRUCTURE operation.

Furthermore, SDE DB2 edit and browse options may be specified on CREATE STRUCTURE and so are saved in the RTO. These options are used by default when the (SDO) structure is used to edit the table data.

`FROM DB2 < (ssn) >`
Indicates that structure generated is derived from a DB2 results table. *ssn* is optional and identifies the local DB2 sub-system name to which a connection will be made.

Before a connection can be made to the DB2 sub-system, the SELCOPY/i DB2 plan must have been bound to that sub-system.

Default for *ssn* is the users default DB2 sub-system name as set by the DB2 Primary Options menu.

table_name
A DB2 base table name, specified as *table_owner.table_name*, containing the columns that constitute the structure's only record type (RTO) definition.

view_name
A DB2 table view, specified as *view_owner.view_name*, which references an SQL query used to generate a results table containing the columns that constitute the structure's only record type (RTO) definition.

DB2 Opts

The following DB2 options are saved in the structure and are used to generate the dynamic SQL query statement, apply further mapping to DB2 column data and also set default values for SDE DB2 table edit. These options are invoked if the structure is applied on edit or browse via the USING STRUCTNAME parameter.

`SELECT (select-clause)`
Specifies a DB2 SQL SELECT clause to be included in the prepared SQL select statement. See "*DB2 SQL Reference*" for syntax of the *select-clause*.

`WHERE (where-clause)`
Specifies a DB2 SQL WHERE clause to be included in the prepared SQL select statement. See "*DB2 SQL Reference*" for syntax of the *where-clause*.

`SORT|ORDER BY (order-by-clause)`
Specifies a DB2 SQL ORDER BY clause to be included in the prepared SQL select statement. See "*DB2 SQL Reference*" for syntax of the *order-by-clause*.

`SKIPLOCKED`
Saves an SDE DB2 table edit default for option SKIPLOCKED.
Ignored unless an isolation level of Cursor Stability (CS) or Read Stability (RS) is in effect, SKIPLOCKED specifies that any selected rows that are already locked by another process should be skipped and not be included in the edit display. See "*DB2 SQL Reference*" for details on the SKIP LOCKED DATA clause.
Default is to allow display of locked rows whenever possible.

`PROTECTPRIMEKEY | EDITPRIMEKEY`
Saves an SDE DB2 table edit default for option PROTECTPRIMEKEY / EDITPRIMEKEY.
Specifies whether data occupying columns that comprise the table's primary key is eligible for update (EDITPRIMEKEY) or is read only (PROTECTPRIMEKEY).
Default is PROTECTPRIMEKEY.

`WITH CS|UR|RR|RS  < KEEP  EXCLUSIVE|UPDATE|SHR >`
Saves an SDE DB2 table edit default for option WITH.
Specifies a DB2 SQL isolation-clause to be included in the prepared SQL select statement. See "*DB2 SQL Reference*" for details on the *isolation-clause* and "*DB2 Performance Monitoring and Tuning Guide*" for details on the effects of isolation level on concurrency and protection of DB2 table data.

The KEEP sub-parameter options are applicable to isolation levels RR (Repeatable Read) and RS (Read Stability) only.
Default isolation level (as set by the SELCOPY/i DB2 package and plan BIND) is CS.

`COMMITONCLEANSAVE | COMMITONSAVE | COMMITONEXIT`
Saves an SDE DB2 table edit default for option COMMITONCLEANSAVE / COMMITONSAVE / COMMITONEXIT.
Specifies when a COMMIT should be executed for changed data.

`COMMITONCLEANSAVE`	COMMIT only if SAVE is executed without errors.
`COMMITONSAVE`	COMMIT on SAVE regardless of errors.
`COMMITONEXIT`	COMMIT only on exit of the edit session.

Default is COMMITONCLEANSAVE.

`COMMITONLOAD`
Saves an SDE DB2 table edit default for option COMMITONLOAD.
Performs a COMMIT following the initial load of rows to be edited, thus releasing all DB2 table locks performed during load of the data. This includes any explicit table locks applied via the LOCKTABLE parameter.

Default is not to perform a COMMIT following load of the table rows.

`AUDIT`
Saves an SDE DB2 table edit default for option AUDIT.
AUDIT will open a new SELCOPY/i DB2 audit data set to record changes to the edited table made during this edit session. See DB2 Auditing for details.
Default is not to perform edit auditing.

Direct Definition
Direct definition syntax provides the method of directly defining an SDO containing one or more record structures (RTOs), each comprising one or more fields, without first having a copy book.

`NAMES`
Field names specified in the *record_definition* are verified so that they conform with the naming standards of the specified programming or query language. Supported standards are as follow:

`ASM | ASSEMBLER | HLASM`	IBM High Level Assembler.
`COBOL`	IBM COBOL.
`DB2`	IBM DB2 SQL.
`PL1 | PLI`	IBM Enterprise PL/1.

Record Definition
Definition of a single record structure (RTO) within the structure definition object (SDO).

`record_type`
The record type name used to identify the new record type object (RTO).

`STRUCTURE`
The data type of an RTO. An RTO may be considered to be a single field of type STRUCTURE which itself comprises a number of individual fields.

`EBCDIC`
`ASCII`
Unless overridden by EBCDIC or ASCII specified as a parameter on a field definition, all character fields within the RTO are to be treated as either EBCDIC or ASCII.
Default is EBCDIC.

`USE WHEN` *`expression`*
Often an individual record can be assigned a record type by matching on record length. Where several record types are potential matches based on record length, then **USE WHEN** should be used to specify criteria that must be satisfied before the record type can be assigned to a particular record.

expression is a valid SDE expression which supports function calls, *record_type* field names and references, sub-expressions, arithmetic, relational and logical operators. The result of the WHEN expression must be numeric and is treated as being Boolean in nature with a zero value indicating a "false" condition and any non-zero value indicating a "true" condition.

If a WHEN expression associated with a record type returns a "true" result for a data record, then that record type is assigned to the record.

Field Definition
Definition of a single field within the record RTO.

`field_name`
The field name identifier.

`PICture` *`picture_string`*
Any COBOL-style picture string. e.g. `PIC S9(7)`
This is treated **purely as comment** data.

`DIMensions`
Used to define an array of given datatype.

dim	An integer that defines the fixed number of elements.
min	An integer that defines the minimum number of elements.
max	An integer that defines the maximum number of elements.
d_name	The field name that defines the variable number of elements.

e.g.

```
dim(12)
dim((0,44,ObjNum))
dim((0,10,OutPrtNum),(0,10,OutPunNum))
```

`EBCDIC`
`ASCII`

For character fields only, treat as either EBCDIC or ASCII.
Default is as defined on the Record Definition, otherwise EBCDIC.

`SIGNED`
`UNSIGNED`
For numeric fields only, treat as signed or unsigned. Default is SIGNED.

The sub-parameters **LEADing|TRAILing** and **SEParate|INCluded** are applicable to zoned decimal only and correspond directly to the COBOL SIGN clause.

`ALIGNED`
`UNALIGNED`
Determines whether the field will be boundary aligned for its data type.
Default is UNALIGNED.

`DEFault` *`default_value`*
Specifies the default value for this field should a new record be added. It can be defined either as a numeric or quoted string constant, or as an expression referring to field names within the current record.

`REMarks` *`remark`*
A comment string placed in either single or double quotes.

`ENUMeration`
Create an enumeration definition or specify the name of an existing enumeration for use with the current field.

enam may be specified to allocate a name to the enumeration definition or as a reference to an enumeration already defined within the current field definition clause.

Each *enumerator* should be specified in the form *display*=*value*, where *display* will be displayed when this field contains *value*. e.g.

```
enum(HLAsm=16,Cobol=17,Pl1=40)
```

display should be enclosed in quotes if it is to contain blanks. *display* may be specified without a corresponding *value*, in which case the previous value plus 1 will be implied.

Data Definition
Data type of a single field within the record RTO.

`BINTeger`
Defines a field to be interpreted as binary whole-number occupying *n_bits* bits, where *n_bits* defaults to **1**.

`BIT`
Defines a field to be interpreted as a series of ON|OFF (1|0) switches occupying *n_bits* bits, where *n_bits* defaults to **1**.

`CHARacter`
Defines a field to be interpreted as a fixed-length character string occupying *n_bytes* bytes, where *n_bytes* defaults to **1**.

`CHARVarying`
Defines a field to be interpreted as a variable-length character string padded with blanks to occupy a fixed *n_bytes*, where *n_bytes* defaults to **1**. The actual length of the string is defined by a 2 byte binary integer at the start of the field so that the overall length of the field is 2+*n_bytes*.
CHARVARYING is equivalent to a PL/1 field which is declared as being CHARACTER VARYING.

`CHARZ`
Defines a field to be interpreted as a variable-length character string occupying a fixed *n_bytes* bytes, where *n_bytes* defaults to **1**. No length field exists, instead the character string is terminated by a null character (x'00') which increases the overall length of the field by 1. (i.e. *n_bytes*+1)
CHARZ is equivalent to a PL/1 field which is declared as being CHARACTER VARYINGZ.

`DECimal`
Defines a field to be interpreted as a packed-decimal number allowing *precision* number of digits in total, of which *scale* number of digits will be displayed following the decimal-point.
Default for *precision* is **7**.
Default for *scale* is **0**.

`DATE`
Defines a date field with the source data interpreted in one of the following formats:

`DECIMAL`	Packed decimal format of length 4 bytes (X'ccyy,dddF'). Decimal values for year of century (yy) and Julian day of year (ddd). 'F' is a 4-bit sign and 'cc' is the century indicator X'00'(1900) or X'01'(2000). e.g. 14th February 2011 is X'0111,045F'.
`BINARY`	Binary format of length 6 bytes (X'yyyy,00mm,00dd'). Binary values for year including century (yyyy), month of year (mm) and day of month (dd). e.g. 20th October 2011 is X'07DB,000A,0014'.
`CATALOG`	ICF Catalog Date format of length 4 bytes (X'yydd,dFcc'). Decimal values for year of century (yy) and Julian day of year (ddd). 'F' is a 4-bit sign and 'cc' is the century indicator X'00'(1900) or X'01'(2000). e.g. 16th July 2008 is X'0819,7F01'.

VTOC	VTOC format of length 3 bytes (X'yydddd'). Binary values for years since 1900 (yy) and Julian day of year (dddd). e.g. 19th August 2005 is X'6900E7'.

Default date format is **DECIMAL**.

FIXed
Defines a field to be interpreted as a binary number allowing *precision* number of digits in total, of which *scale* number of digits will be displayed following the decimal-point.
Default for *precision* is **7**.
Default for *scale* is **0**.

FLOATBIN
Defines a field to be interpreted as a Binary (IEEE 754) Floating-point number occupying *n_bytes* bytes, where *n_bytes* may be 4 or 8 (defaults to **4**).

FLOATDEC
Defines a field to be interpreted as a Decimal Floating-point value of *precision* number of significant digits, where *precision* may be 16 or 34 (defaults to **34**). Precision 16 occupies 8 bytes of data, precision 34 occupies 16 bytes.

FLOATHEX
Defines a field to be interpreted as a Hexadecimal Floating-point number occupying *n_bytes* bytes, where *n_bytes* may be 4 or 8 (defaults to **4**).

HEXadecimal
Defines a field to be interpreted as printable hexadecimal character string occupying *n_bytes* bytes, where *n_bytes* defaults to **1**.

INTeger
Defines a field to be interpreted as a binary whole-number occupying *n_bytes* bytes, where *n_bytes* defaults to **4**.

PCHAR
Defines a field to be interpreted as a PL/1 style PICTURE string representing a character data item (i.e. no numerical interpretation). The length of the field is determined by the quoted *pl1_picture_string*.
pl1_picture_string may contain any valid PL/1 picture character for character data. If invalid characters (e.g. numeric picture characters E, V, Z, etc.) are specified then an error will occur. Default is PCHAR('X').

PFIXED
Defines a field to be interpreted as a PL/1 style PICTURE string representing a FIXED point numerical character data item. The length of the field is determined by the quoted *pl1_picture_string*.
pl1_picture_string may contain any valid PL/1 picture character for numeric character data except for exponent characters "E" and "K". If invalid characters are specified then an error will occur. Default is PFIXED('S9').

PFLOAT
Defines a field to be interpreted as a PL/1 style PICTURE string representing a FLOAT point numerical character data item. The length of the field is determined by the quoted *pl1_picture_string*.
pl1_picture_string may contain any valid PL/1 picture character for numeric character data. If invalid characters are specified then an error will occur. Default is PFLOAT('S9ES99').

STRUCTure
Defines a field to be interpreted as a structure containing one or more fields specified using Field Definition syntax. One or more of these fields may themselves be structures and so nesting of structures is supported. Each field definition should be separated by a comma.

TIME
Defines a time field with the source data interpreted in one of the following formats:

DECIMAL	Packed decimal format of length 4 bytes (X'HHMM,SSTH'). Decimal values for hours (HH), minutes (MM), seconds (SS), tenths of second (T) and hundredths of second (H). Note that the data is unsigned.
BINARY	Binary format of length 4 bytes (X'nnnn,nnnn'). A 32-bit unsigned binary value for number of hundredths of seconds (0.01 second) elapsed in a day.

Default time format is **DECIMAL**.

TIMESTAMP
Defines a timestamp field with the source data interpreted in one of the following formats:

DECIMAL	Packed decimal format of length 8 bytes (X'yyyy,dddF,HHMM,SSTH'). See descriptions of DECIMAL format for DATE and TIME parameters.
BINARY	Binary format of length 10 bytes (X'yyyy,00mm,00dd,nnnn,nnnn'). See descriptions of BINARY format for DATE and TIME parameters.
STCK	Store Clock value of length 8 bytes. A 64-bit unsigned binary value in the same format as the system TOD clock.
HFSDIR	HFS directory timestamp of length 4 bytes. A 32-bit unsigned binary value for number of seconds elapsed since 1970/01/01 00:00:00.

Default timestamp format is **DECIMAL**.

`UNION`
Defines a field to be interpreted as a union of one or more fields specified using Field Definition syntax. Each field definition should be separated by a comma.
Unions allow the same data to be interpreted as more than one data-type.

`VARCHAR`
Defines a field to be interpreted as a variable-length character string. The maximum possible length of the string is defined as **max_bytes** bytes, defaulting to **0**. The actual length of the string is defined by a **len_bytes** bytes binary integer at the start of the field.
Default for *len_bytes* is **2**. The value of the *exclusive* parameter indicates whether this length integer includes or exlcudes the length field itself. Permissable values are EXCLUSIVE (the default) and INCLUSIVE.

`XVARCHAR`
Defines a field to be interpreted as a variable-length character string. The maximum possible length of the string is defined as *max_bytes* bytes, defaulting to **0**. The actual length of the string is defined by field name *len_field*, which must be numeric.

`ZONEd`
Defines a field to be interpreted as a zoned-decimal character number allowing *precision* number of digits in total, of which *scale* number of digits will be displayed following the decimal-point.
Default for *precision* is **1**.
Default for *scale* is **0**.

Examples:

```
<sd create structure          CBL.CBLI.STRUCT(EMP) from cobol CBL.COPYBOOK.COBOL(EMP)
<sd edit  CBL.SDE.EMP   using CBL.CBLI.STRUCT(EMP)
```

Sample commands as they may appear in the user's HOME CMX file. The first command will generate a new SDO structure from a single COBOL copy book, the second will edit a file using the SDO to apply record mapping.

```
<sd create structure   CBL.CBLI.SDO(FILSTRUC)        \
  (                                                  \
    REC-TYPE01 struct                                \
    (  REC-TYPE              char(2),                \
       NAME                  char(20),               \
       EMPLOYEE-NO           int(2) unsigned,        \
       AGE                   int(2) unsigned,        \
       SALARY                dec(7),                 \
       MONTH                 int(4) dim(12) unsigned,\
                             char(2)                 \
    )  use when REC-TYPE='01'                        \
   ,                                                 \
    REC-TYPE02 struct                                \
    (  REC-TYPE              char(2),                \
       NAME                  char(20),               \
       JOB-TITLE             char(14),               \
       ADDR1                 char(20),               \
       ADDR2                 char(20),               \
       POSTCODE              char(4)                 \
    )  use when REC-TYPE='02'                        \
  )                                                  \
  names(COBOL)     replace    case respect
```

Sample command as it may appear in the user's HOME CMX file. This command will generate a new SDO structure consisting of 2 record type definintions.

```
<||sd create structure CBL.CBLI.SDO(FILSTRUC) \
  (                                                \
    REC-TYPE01 struct                              \
    (  REC-TYPE              char(2),              \
       NAME                  char(20),             \
       EMPLOYEE-NO           int(2) unsigned,      \
       AGE                   int(2) unsigned,      \
       SALARY                dec(7),               \
       MONTH                 int(4) dim(12) unsigned, \
                             char(2)               \
    )  use when  REC-TYPE='01'                     \
   ,                                               \
    REC-TYPE02 struct                              \
    (  REC-TYPE              char(2),              \
       NAME                  char(20),             \
       JOB-TITLE             char(14),             \
       ADDR1                 char(20),             \
       ADDR2                 char(20),             \
       POSTCODE              char(4)               \
    )  use when (                                  \
                    (  REC-TYPE < '02'             \
                     & REC-TYPE > '05'             \
                    )                              \
                  |                                \
                    (  REC-TYPE < '22'             \
                     & REC-TYPE > '25'             \
                    )                              \
                )                                  \
  )                                                \
  names(COBOL)     replace    case respect
```

Sample command as it may appear in the user's HOME CMX file. This command will generate a new SDO structure consisting of 2 record type definintions. Additional USE WHEN criteria is specified to identify the circumstances under which the particular record mapping is to be used.

See Also:

EDIT
BROWSE

CURSOR

Syntax:

```
>>-- CURsor --+- HOME ---------------+----------------------------------------><
              |                      |
              +- CMdline ------------+
```

Description:

Position the cursor within the current SDE window view. The CURSOR command is most often used in macros.

Note: The CURSOR command does not alter the window view itself. i.e. the current line and column remains unchanged.

Parameters:

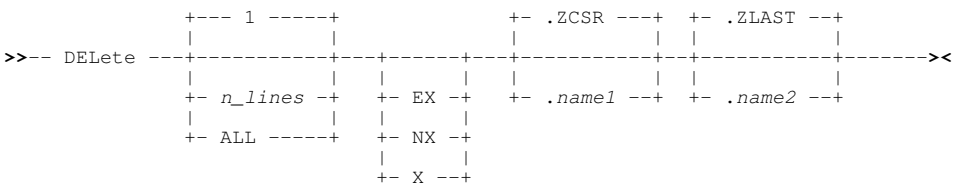
HOME
: If the cursor is on the command line, the cursor is positioned at the line and column at which it was positioned when it was last in the display area. If this line is no longer visible within the current SDE window view, the cursor is positioned in column 1 of the current line.

  If the cursor is in the file area, the cursor is positioned at column 1 of the command line.

CMDLINE
: The cursor is positioned at column 1 of the command line.

DELETE

Syntax:



Description:

Delete visible data records (not excluded) and/or EXCLUDED groups of records.

Deletion of records is not allowed for Update-in-place editing unless the data set is an RRDS or VRDS. For these types of files, deleted records become empty slots.

A group of EXCLUDED records corresponds to a single line for deletion regardless of the number of excluded lines it represents or whether excluded shadow lines are set on or off. (See SET SHADOW.)

DELETE is not sensitive to record type and so may delete records of different record types as part of the same execution.

Records groups that are NOTSELECTED or are SUPPRESSED are not eligible for deletion. Similarly, if EXCLUDED shadow lines are set off, EXCLUDED record groups are also not eligible for deletion unless parameter EX or X is specified.

Also see the "D(n)" and "DD" delete prefix area commands.

Parameters:

n_lines
: Selects a number of lines from the top of the range upon which the DELETE operation will apply.

  The *n_lines* value includes visible records and EXCLUDED, SUPPRESSED and NOTSELECTED shadow lines. Each shadow line corresponds to a single line in the *n_lines* line count, regardless of the number or records within the record group which it represents.

  In general, if shadow lines set off, then they are **not** included in the *n_lines* line count. The exception to this is when EX or X is specified, in which case EXCLUDED shadow lines are always included within the *n_lines* line count, regardless of whether excluded shadow lines are set on or off.

  Only lines that are eligible for deletion and are included within the lines selected by *n_lines*, will be deleted. e.g. SUPPRESSED shadow lines may be included within the selected lines but are not eligible for deletion and are, therefore, not deleted.

  The default value of *n_lines* is 1.

ALL
: All lines within the specified or implied range are selected for DELETE.

EX
X
: Only EXCLUDED record groups are eligible for deletion. Visible data records are not eligible.

  Only when EX or X is used will all EXCLUDED record groups that fall within the selected number of records, be deleted regardless of whether excluded shadow lines are set on or off.

  Default is that both EXCLUDED record groups and visible data records are eligible for delete, in which case, if excluded shadow lines are set off, EXCLUDED record groups are not eligible for delete.

NX
: Only visible data records are eligible for deletion. EXCLUDED record groups are not eligible.
  Default is that both EXCLUDED record groups and visible data records are eligible for delete.

.name1
: A label name identifying the first record of a range of data records to be deleted. The preceding "." (dot) is mandatory.
  Default is .ZCSR (the focus line.)

.name2
: A label name identifying the last record of a range of data records to be deleted. The preceding "." (dot) is mandatory.
  If *.name2* occurs before *.name1* in the display, then the order is reversed.
  The default is .ZLAST.

Examples:

`del 3`
: Delete all eligible lines within the first 3 lines of data starting at the focus line. Eligible lines are visible data records and shadow lines representing an EXCLUDED record group.

`delete nx`
: Delete the focus line if the focus line is a visible data record.

`del all x .fix .fixe`
: Delete all EXCLUDED record groups that exist within the range of lines starting at label name .FIX, ending at label name .FIXE. (The setting for excluded shadow lines is irrelevant.)

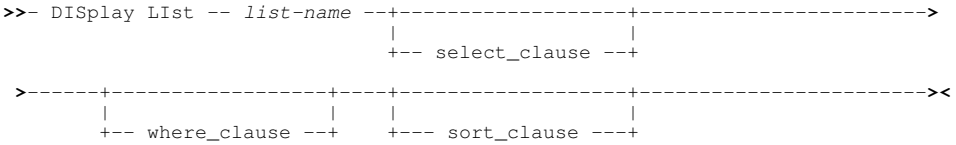
`del 20 ex`
: Delete all EXCLUDED record groups that exist within the first 20 lines of data starting at the focus line. Each EXCLUDED record group constitutes a single line regardless of whether excluded shadow lines are set on or off. Therefore, even with excluded shadow lines set off, 10 consecutive excluded records are treated as a single line for deletion.

See Also:

INSERT

DISPLAY LIST

Syntax:



Description:

Open a SELCOPY/i List window to display an in-storage list generated via the CREATE LIST command.

The List window displaying the list data will support most of the standard list window features as follow:

- Field Descriptor Block (FDB)
- Edit View
- Selecting, Sorting and Filtering
- Sorting with the Cursor

On exiting the List window display, the in-storage list is destroyed and storage is freed.

Parameters:

list_name
: The name of the in-storage list data to be displayed.

select_clause
: Identifies a valid SELECT Clause to display only those selected columns.

where_clause
: Identifies a valid WHERE Clause to filter rows of data.

sort_clause
: Identifies a valid SORT Clause to sort the specified columns of data.

Examples:

`>sd display list AMLIST select KEY,COMPNAME,FIRSTNAME,LASTNAME,DEPT where lastname=JONES sort DEPT a`
: Open a list window for the in-storage list AMLIST with select, sort and filter clauses applied.

See Also:

CREATE LIST

DISPLAY RECTYPES

Syntax:

```
>>-+- DISplay RECTypes --+--+---------------+--------------------------------><
   |                     |  |               |
   +- LR ----------------+  +- struct_name -+
```

Description:

Open a SELCOPY/i List window to display each record_type in the specified structure definition object (SDO).

Parameters:

struct_name
The name of the SDO structure.
Default is the name of the SDO structure used in the current SDE window.

Examples:

`display rectypes NBJ2.CBLI.SDO(COBSALES)`
Open a record type list window for the structure definition file NBJ2.CBLI.SDO(COBSALES).

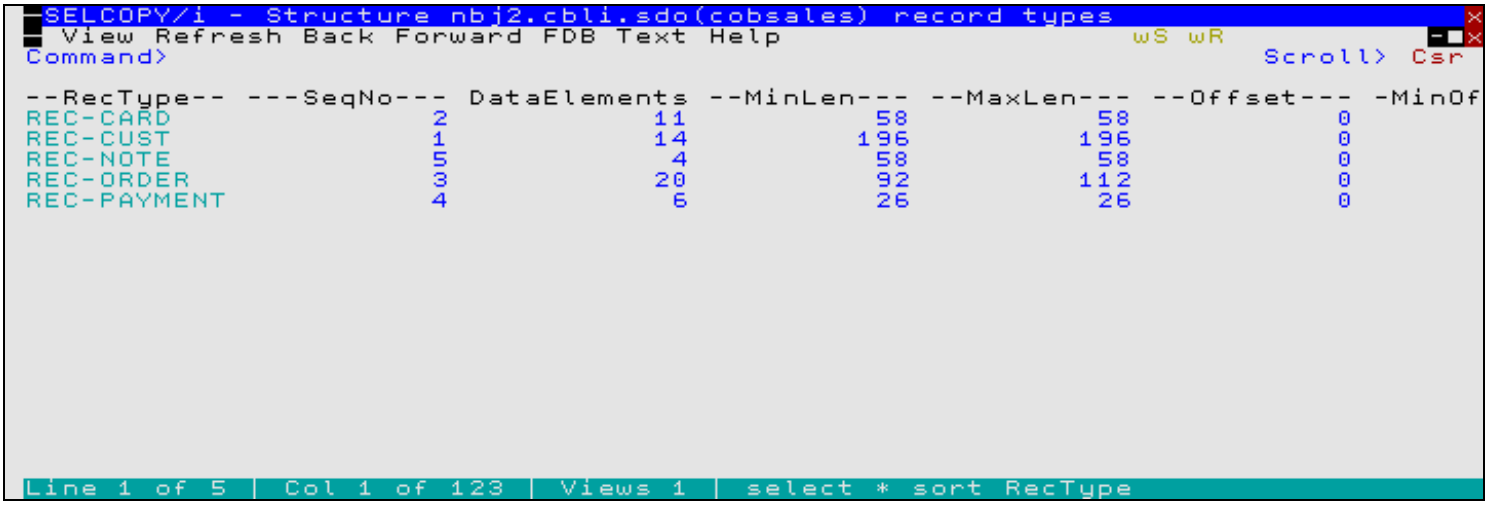

SELCOPY/i - Structure nbj2.cbli.sdo(cobsales) record types
View Refresh Back Forward FDB Text Help
wS wR
Command>
Scroll> Csr
--RecType-- ---SeqNo--- DataElements --MinLen--- --MaxLen--- --Offset--- -MinOf
REC-CARD 2 11 58 58 0
REC-CUST 1 14 196 196 0
REC-NOTE 5 4 58 58 0
REC-ORDER 3 20 92 112 0
REC-PAYMENT 4 6 26 26 0
Line 1 of 5 | Col 1 of 123 | Views 1 | select * sort RecType

Figure 39. Display Record Type List View.

Columns Displayed:

Name	Type	Description
RecType	ALPair	Record type
SeqNo	Int	Record type sequence number
DataElements	Int	Number of Data Elements
MinLen	Int	Minimum length
MaxLen	Int	Maximum length
Offset	Int	Offset
MinOffset	Int	Minimum Offset
UseWhen	ALPair	Record type identification criteria
Flag1	Hex	RTO flags
Flag2	Hex	RTO flags
UseNever	BitFlag	This record-type mapping is NEVER to be applied.
UseAlways	BitFlag	This record-type mapping is ALWAYS to be applied.

DISPLAY STRUCTURE

Syntax:

```
>>-+- DISplay STRUCTure -+--+---------------+--------------------------------><
   |                    |  |               |
   +- LS ----------------+  +- struct_name -+
```

Description:

Open a SELCOPY/i List window to display the field definitions for each record_type in the specified structure definition object (SDO).

The List window displaying supports the standard list window features as follow:

- Field Descriptor Block (FDB)
- Edit View
- Selecting, Sorting and Filtering
- Sorting with the Cursor

Parameters:

struct_name
The name of the SDO structure.
Default is the name of the SDO structure used in the current SDE window.

Examples:

`display structure NBJ2.CBLI.SDO(COBSALESC)`
Open a list window for the structure definition PDSE member NBJ2.CBLI.SDO(COBSALES).

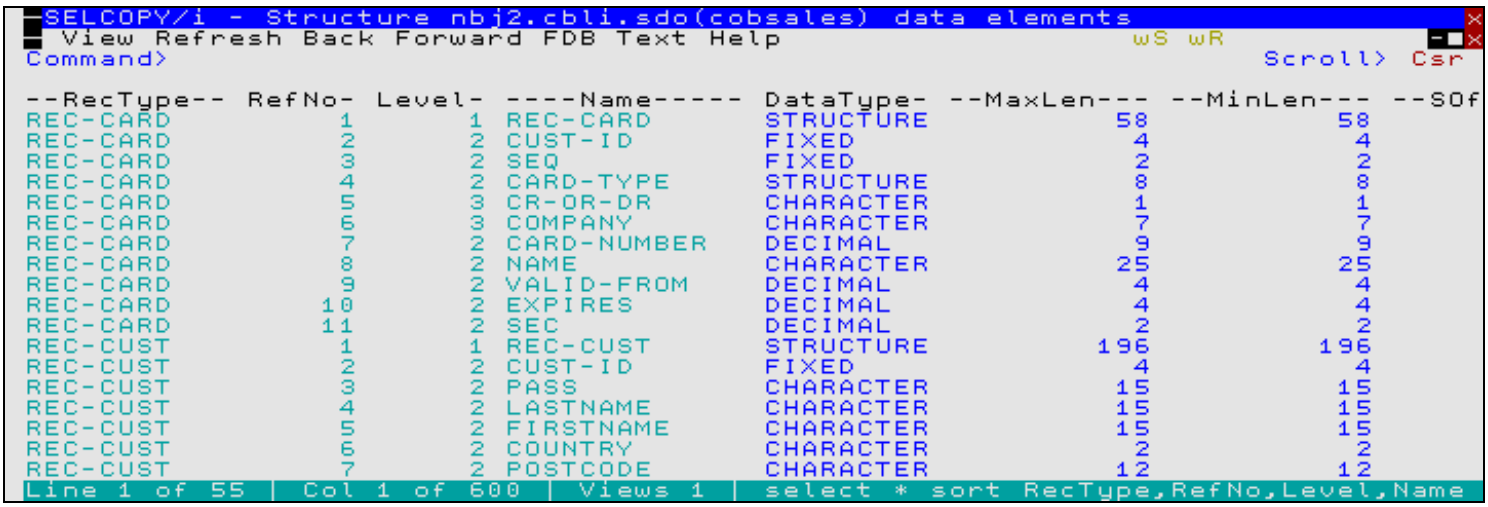

SELCOPY/i - Structure nbj2.cbli.sdo(cobsales) data elements
View Refresh Back Forward FDB Text Help
wS wR
Command>
Scroll> Csr
--RecType-- RefNo- Level- ----Name----- DataType- --MaxLen--- --MinLen--- --SOf
REC-CARD 1 1 REC-CARD STRUCTURE 58 58
REC-CARD 2 2 CUST-ID FIXED 4 4
REC-CARD 3 2 SEQ FIXED 2 2
REC-CARD 4 2 CARD-TYPE STRUCTURE 8 8
REC-CARD 5 3 CR-OR-DR CHARACTER 1 1
REC-CARD 6 3 COMPANY CHARACTER 7 7
REC-CARD 7 2 CARD-NUMBER DECIMAL 9 9
REC-CARD 8 2 NAME CHARACTER 25 25
REC-CARD 9 2 VALID-FROM DECIMAL 4 4
REC-CARD 10 2 EXPIRES DECIMAL 4 4
REC-CARD 11 2 SEC DECIMAL 2 2
REC-CUST 1 1 REC-CUST STRUCTURE 196 196
REC-CUST 2 2 CUST-ID FIXED 4 4
REC-CUST 3 2 PASS CHARACTER 15 15
REC-CUST 4 2 LASTNAME CHARACTER 15 15
REC-CUST 5 2 FIRSTNAME CHARACTER 15 15
REC-CUST 6 2 COUNTRY CHARACTER 2 2
REC-CUST 7 2 POSTCODE CHARACTER 12 12
Line 1 of 55 | Col 1 of 600 | Views 1 | select * sort RecType,RefNo,Level,Name

Figure 40. Display Structure List View.

Columns Displayed:

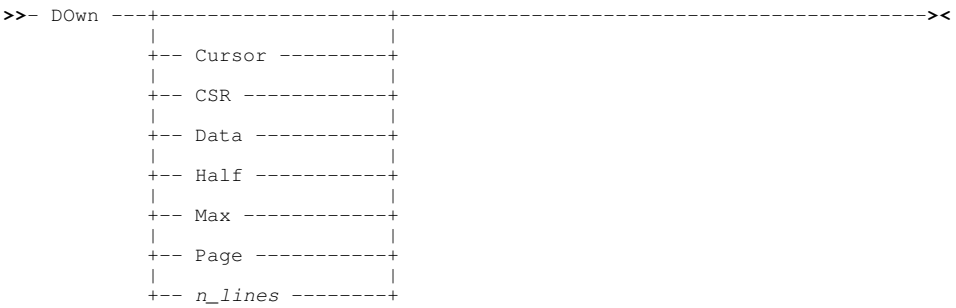
Name	Type	Description
RecType	ALPair	Record type
RefNo	Int	Reference number
Level	Int	Level
Name	ALPair	Name
DataType	ALPair	Data type
MaxLen	Int	Maximum length
MinLen	Int	Minimum length
SOffset	Int	Offset in structure
POffset	Int	Offset in parent

BitOff	Int	Bit offset
FType	Char	Field content type
Precision	Int	Precision
Scale	Int	Scale
Picture	ALPair	Picture
Dims	Int	Total dimensions
Dim	Int	Dimension at current level
MaxArray	Int	Maximum number of array elements
MaxIV	Int	Maximum integer value of reference
MinIV	Int	Minimum integer value of reference
Flags1	Hex	Flags
VarLen	BitFlag	Variable inherent length
VarDim	BitFlag	Occurs in array of variable dimension
VarSOff	BitFlag	Offset in structure is variable
VarPOff	BitFlag	Offset in parent is variable
HasVarOff	BitFlag	Group has elements with variable offsets
VarRef	BitFlag	Referred to in size or dimension
UFlags	Hex	Usage flags
Aligned	BitFlag	Aligned
Signed	BitFlag	Signed
Filler	BitFlag	Unnamed
DFlags	Hex	Display flags
LZeros	BitFlag	Display with leading zeros
ASCII	BitFlag	Display with ASCII to EBCDIC translation
Num	Enum	Numeric Type
CPMax	Int	Maximum number of create parameters
CPAct	Int	Actual number of create parameters
CPType01	Enum	Create parameter type 01
CPIntV01	Int	Create parameter integer value 01
CPChar01	ALPair	Create parameter character value 01
CPType02	Enum	Create parameter type 02
CPIntV02	Int	Create parameter integer value 02
CPChar02	ALPair	Create parameter character value 02
CPType03	Enum	Create parameter type 03
CPIntV03	Int	Create parameter integer value 03
CPChar03	ALPair	Create parameter character value 03
CPType04	Enum	Create parameter type 04
CPIntV04	Int	Create parameter integer value 04
CPChar04	ALPair	Create parameter character value 04
DimMax01	Int	Maximum dimension 01
DimMin01	Int	Minimum dimension 01
DimExp01	ALPair	Dimension expression 01
DimMax02	Int	Maximum dimension 02
DimMin02	Int	Minimum dimension 02
DimExp02	ALPair	Dimension expression 02
DimMax03	Int	Maximum dimension 03
DimMin03	Int	Minimum dimension 03

DimExp03	ALPair	Dimension expression 03
DimMax04	Int	Maximum dimension 04
DimMin04	Int	Minimum dimension 04
DimExp04	ALPair	Dimension expression 04
DimMax05	Int	Maximum dimension 05
DimMin05	Int	Minimum dimension 05
DimExp05	ALPair	Dimension expression 05
Rem	ALPair	Remarks string

DOWN

Syntax:



Description:

Scroll the view of the data within the SDE window downwards towards the bottom of the file data.

Where no parameter is specified, the scroll amount will be the value specified in the **Scroll>** field in the top right corner of the window display.

Note that the first data record in any multi record view will **always** be preceded by its complete group of column header lines.

For single record views, the standard headers are always visible at the top of the display area and are not included as part of the scrollable text. Each non-header line within a single record view is identified as being a **field data line**.

Where the cursor is positioned at an offset within a column header line of a multi record view, the cursor is deemed to be located at the same offset within the data record that immediately follows the group of header lines. Therefore, DOWN CURSOR will operate successfully when the cursor is positioned within a header line.

UP and DOWN scroll commands will cause the window display area to be adjusted by a number of lines determined by the number of multi record view data records and shadow lines, or single record view field data lines. Other lines in the display area, i.e. Header lines and blank filler lines that precede a group of header lines or follow the "End of Data" record, are not included in this calculation.

Attempting to scroll down beyond the last entry (data record, shadow line or field data line) will make the "End of Data" record the first line of the scrolled display.

Parameters:

CURSOR
CSR
: The data record, shadow line or field data line on which the cursor is positioned becomes the first line of the scrolled display.
If the cursor is positioned on a header line, the data record or field data line immediately following the header line becomes the first line in the display area.
If the cursor is positioned outside the display area or on the first line within the display area, then DOWN PAGE is executed instead.

DATA
: Scroll down to display one page (display window depth) less one line of data.
The last data record, shadow line or field data line in the current display area becomes the first line of the scrolled display.

HALF

Scroll down half a page of data.
The data record, shadow line or field data line that is half way down the page of data in the current display area becomes the first record of the scrolled display.

`MAX`
Scroll down to display the last page of data.
Where more than one page of data exists, the "End of Data" line becomes the last line of the scrolled display. Otherwise, the first line of data becomes the first line of the scrolled display.
Equivalent to the BOTTOM command.

`PAGE`
Scroll down to display the next whole page of data.
The data record, shadow line or field data line following the last line of the current display area becomes the first line of the scrolled display.

`n_lines`
Scroll down a specified number of lines.
The data record, shadow line or field data line that is *n_lines* below the current line becomes the first line of the scrolled display.

Examples:

`down 5`
Scroll the display area down 5 lines.

See Also:

LEFT
RIGHT
UP

DROP, DDROP

Syntax:

```
>>- DROP --- struct_name ----------------------------------------------------><

>>- DDROP -- struct_name ----------------------------------------------------><
```

Description:

Drop from storage a Structure Definition Object (SDO) used within SDE, An SDO (file-structure) provides the information required to interpret indiviudal logical fields within a file's data records.

If changes have been made to the structure since it was last saved using SAVESTRUCTURE, (USE WHEN criteria may have been added for instance) then the user will be prompted with a dialog asking whether or not the changes are to be saved.

DDROP removes the SDO without prompting the user to save changes.

Parameters:

`struct_name`
The name of the structure (SDO) being dropped.

See Also:

CREATE STRUCTURE
SAVESTRUCTURE

DUPLICATE

Syntax:

Description:

Duplicate the focus line record a specified number of times.

By default, DUPLICATE operates on both visible data records (not excluded) and EXCLUDED record groups. Records groups that are NOTSELECTED or are SUPPRESSED are not affected.

Parameters:

n_lines
Number of times the focus line is to be duplicated.
The default is 1 line.

Examples:

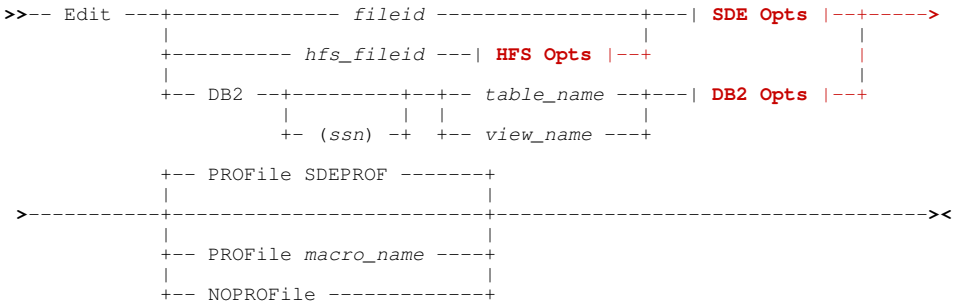
`dup 3`
Duplicate the focus line 3 times.

See Also:

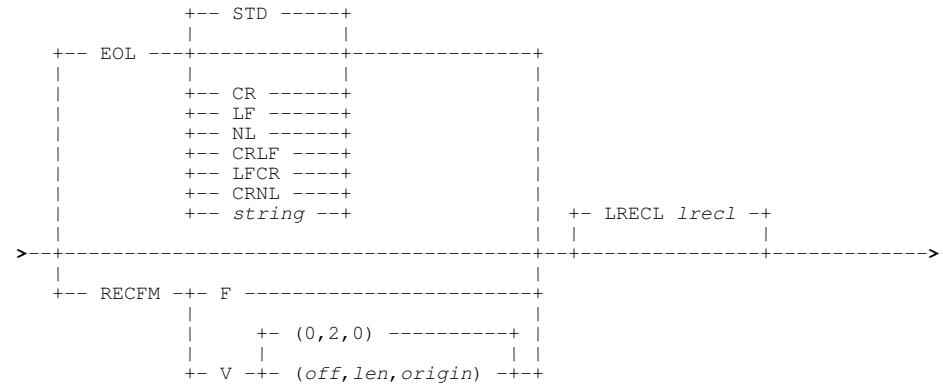
INSERT
DELETE

EDIT

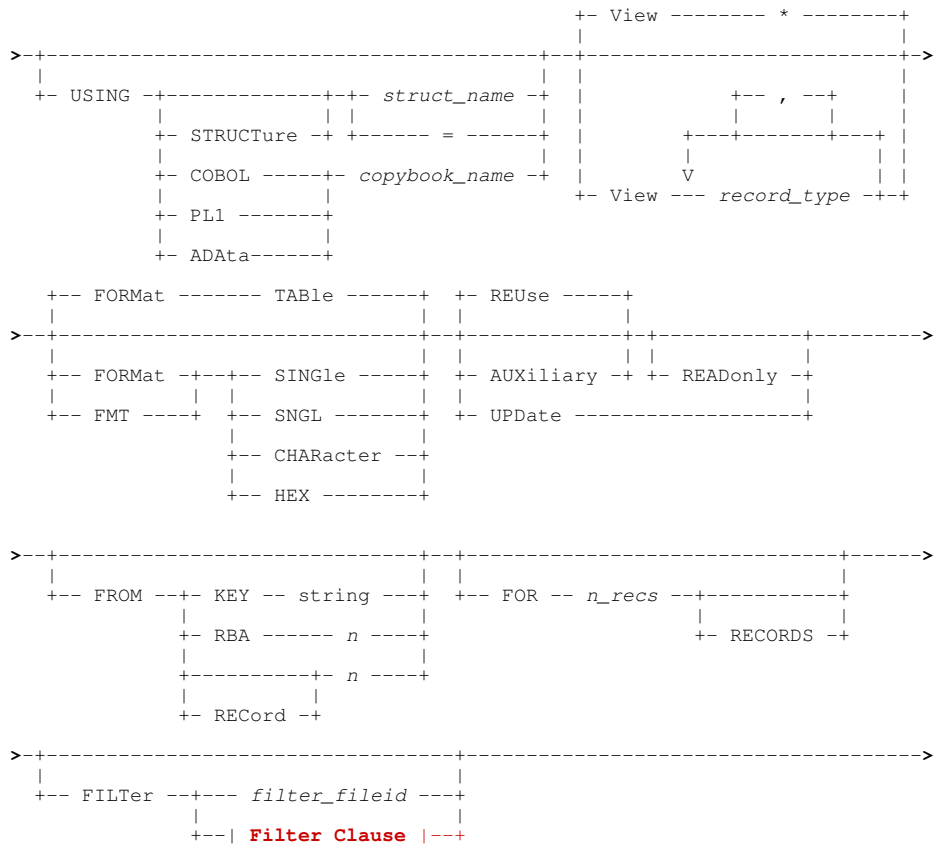
Syntax:


>>-- Edit ---+-------------- fileid ------------------+---| SDE Opts |--+----->
| | |
+---------- hfs_fileid ---| HFS Opts |--+ |
| |
+-- DB2 --+---------+--+-- table_name --+---| DB2 Opts |--+
| | | |
+- (ssn) -+ +-- view_name ---+
+-- PROFile SDEPROF -------+
| |
>-----------+--------------------------+-----------------------------------><
| |
+-- PROFile macro_name ----+
| |
+-- NOPROFile -------------+

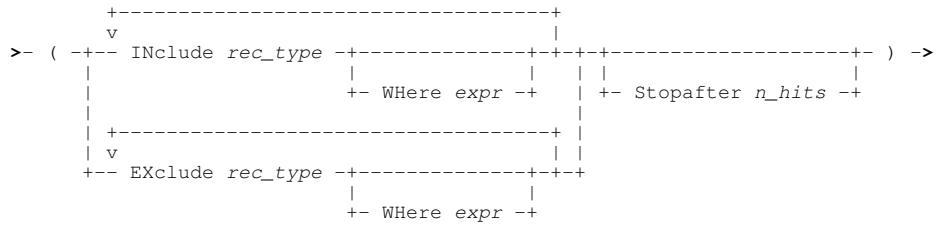
HFS Opts:



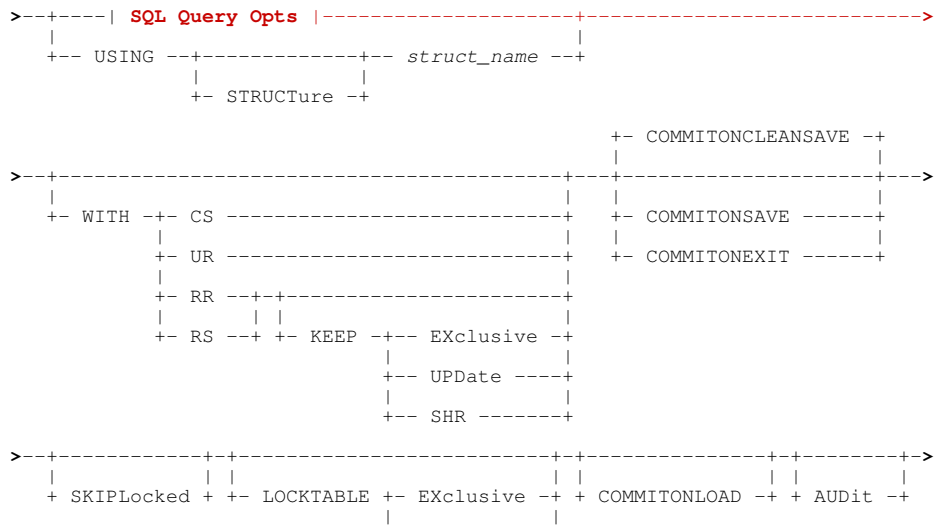
SDE Opts:

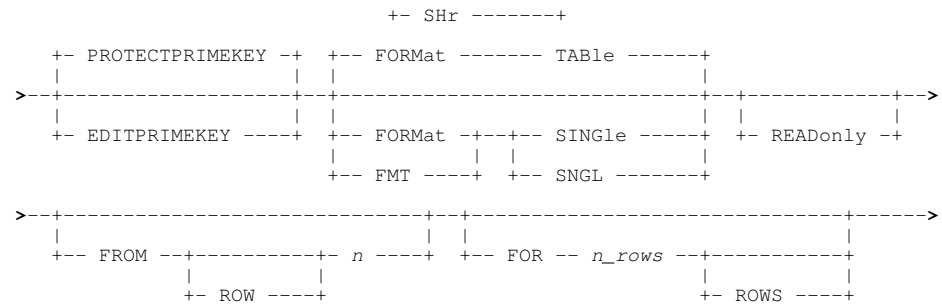

+- View -------- * --------+
USING -+- struct_name -+
+- STRUCTure -+ +------ = ------+
+- COBOL -----+- copybook_name -+
+- PL1 -------+
+- ADAta------+
+- View --- record_type -+-+
+-- , --+
FORMat ------- TABle ------+
+- REUse -----+
FORMat -+--+-- SINGle -----+
+- AUXiliary -+ +- READonly -+
FMT ----+ +-- SNGL -------+
+- UPDate -------------------+
+-- CHARacter --+
+-- HEX --------+
FROM --+- KEY -- string ---+
+-- FOR -- n_recs --+-----------+
+- RBA ------ n ----+
+- RECord -+
FILTer --+--- filter_fileid ---+
+--| Filter Clause |--+

Filter Clause:

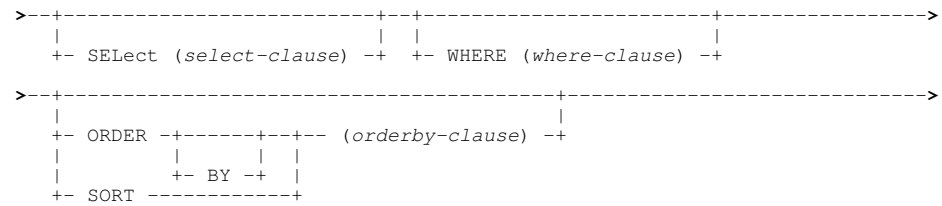


DB2 Opts:


+--| SQL Query Opts |--+
+-- USING --+-- struct_name --+
+- STRUCTure -+
+- COMMITONCLEANSAVE -+
WITH -+- CS ------+
+- COMMITONSAVE ------+
+- UR --------+
+- COMMITONEXIT ------+
+- RR --+-+-+
+- RS --+ +- KEEP -+-- EXclusive -+
+-- UPDate ----+
+-- SHR -------+
+ SKIPLocked + +- LOCKTABLE +- EXclusive -+ + COMMITONLOAD -+ + AUDit -+



SQL Query Opts:



Description:

Edit a structured data set or HFS file, or edit a DB2 results table within SDE.

An SDE edit display window is opened and focus is passed to the new window. Depending on the value on the FORMAT parameter, the SDE display is either a multi record window view or a single record window view for the first record selected.

For both Structured Data Edit alpha-numeric (AN) fields and DB2 Table Edit character (CH or VCHAR) fields that contain non-printable text, only the printable characters may be edited. These editable areas of the character data occupy immoveable fixed positions and lengths within the field which are represented by underscores. The non-printable characters may only be edited by setting HEX ON and updating the hex display.

Structured Data Edit:

If the Structure Definition Object (SDO) specified by the USING STRUCTURE parameter is not already in storage, it is loaded from the appropriate Structure Definition File (SDF). Record Type Objects (RTO) within the specified SDO are then used to map the records in the data set.

For every record in the file, each RTO is checked until one is found whose criteria is matched by the record characteristics. This RTO is then used to map the record. If the record does not match any RTO criteria, then the first RTO in the SDO is used by default. The RTO automatically selected to map a record may be changed by the user via the USE WHEN command or the SDE Edit/Browse Utility Window.

By default, all fields within records that have an associated RTO are displayed as printable character, numeric data fields having first been converted to decimal.

If no SDO is specified, then FORMAT CHARACTER is applied to each record. i.e. the data is displayed as a single, variable (RECFM=V) or fixed (RECFM=F) length character field with field name "UnMapped". No data conversion is performed.

If the VIEW parameter is specified with a non-generic argument, the DRECTYPE setting is initialised as the first *record_type* parameter specified on the VIEW parameter. Otherwise, the DRECTYPE setting is initialised as described in *"Default Record Type"*

With the exception of alpha-numeric (AN) fields, record data that does not meet the specifications of its field definition is considered to be invalid and is represented by asterisks.

DB2 Table Edit:

DB2 table edit is supported for in-storage edit/browse only. Therefore, if a results table is too large to be loaded into available storage a storage error occurs. The user must then restrict the number of rows to be loaded using further row selection criteria (i.e. FROM, FOR, SELECT, WHERE parameters).

On starting a DB2 table edit, a temporary SDO is created reflecting field characteristics obtained from the SQL Descriptor Area return by the prepared SQL SELECT execution. Furthermore, a new DB2 connection is made and, optionally, a new SELCOPY/i DB2 audit log is allocated for each new SDE edited DB2 base or results table. Opening second and subsequent SDE edit views of the same table data does not open a new DB2 connection or audit log data set.

Parameters:

fileid

The fileid of the data set containing records to be edited.
fileid may be the DSN of a sequential or VSAM data set or the DSN and member name of a PDS/PDSE member.

hfs_fileid
A complete HFS (or zFS) file path containing data to be edited.
hfs_fileid is identified as an HFS path if *fileid* is not a valid DSN for a sequential, VSAM or PDS/PDSE dataset. e.g. DSN begins with ".", contains invalid special characters (e.g. "/") or contains qualifiers of length greater than 8.

HFS Opts
The following HFS options may be specified to determine the handling of data within the HFS file.

EOL=STD|NL|CR|LF|CRLF|LFCR|CRNL|*string*
Sets the EOLIN (input end-of-line) delimiter value used to determine the end of each record for non-RECFM F/V input. EOLIN delimiters are not included in the edited record data or record length. EOL parameter elements are as follow:

STD	-	Any standard line-end.
NL	X'15'	New Line.
CR	X'0D'	Carriage Return.
LF	X'0A'	Line Feed.
string	-	A 2-byte user specified character or hex string.

STD is default so that the EDIT operation scans the input data for any of the standard EOL combinations (not *string*), stopping when one is found. This EOL combination is used as EOLIN for the file.

RECFM F | V (*off*,*len*,*origin*)
Specifies that the data is to be treated as containing Fixed or Variable length format records.

RECFM F indicates that all records are of a fixed length as defined by the LRECL argument.

RECFM V allows the user to specify the location of the record length fields within the data as follows:

off	Offset of the record length field from the start of the record.
len	Length of the record length field.
origin	The start of the record data at which the record length is applied.

Default is (0,2,0) which describes standard RECFM V organisation data sets.
The length field will be displayed as part of the data, so, unless editing the data using a suitable associated structure (SDO), the user must take care not to corrupt the length field and also maintain it for any change in record length.

LRECL *lrecl*
Specifies the maximum record length of input records.

Records terminated by an EOL sequence will wrap onto the next line of data if the record length exceeds *lrecl*. Where a record has wrapped, the prefix area contains the "==EOL>" flag. Furthermore, read-only edit is forced in order to suppress save of a wrapped record as multiple, individual records.

For RECFM F data, *lrecl* is the fixed length of the records in the edit view. If the file size is not a multiple of the fixed format lrecl value then, an error occurs and edit is cancelled. For display purposes only, using BROWSE with any *lrecl* value will display the data with the last record padded with blanks up to the *lrecl* length.

If the record length field of a RECFM V record exceeds the *lrecl* value, then an error is returned.

RECFM V and EOL delimited records have default *lrecl* of 32752, wheras RECFM F records have default *lrecl* of 80.

SDE Opts
The following SDE options are applicable to edit of structured data found in HFS files or z/OS data sets only. See DB2 Opts for DB2 table edit options.

USING (STRUCTURE) *struct_name*
Identifies the SDO to be applied to the data records. *struct_name* is the SDF fileid assigned to the SDO in a CREATE STRUCTURE command.

"=" (equals) may be specified instead of *struct_name* to indicate the name of the structure being used in the current SDE window.

USING COBOL | PL1 | ADATA *copybook_name*
Identifies the full fileid of *copybook_name*, a COBOL/PL1 copy book or COBOL/PL1 ADATA output file to be used to format the data records.

Using *copybook_name* syntax has a processing overhead in that a CREATE STRUCTURE operation is performed to generate a temporary SDO. This overhead is greater if COBOL or PL1 is specified since a compile of the copybook is performed to first obtain the ADATA file output. Therefore, it is recommended that a non-temporary SDO is generated using CREATE STRUCTURE and that this is used for future formatting of the

data (on EDIT, BROWSE, FSU, etc.)

`VIEW record_type`
`VIEW *`

Identifies one or more record types for which records will be displayed in the initial SDE view. Records that are associated with other record types are not visible within the display but are displayed as shadow lines instead. * (asterisk) indicates all record types including the internal record type "UnMapped" (or "UnMappedSeg") which is used to map data records that have no associated RTO in the SDO.

Records within the display may be later suppressed or made visible using the VIEW command.

Default is VIEW *.

`FORMAT`
`FMT`

Specifies the format in which record data will be displayed in the initial SDE view.

`SINGLE` `SNGL`	Single record format.
`TABLE`	Table format. (Default)
`CHARACTER`	Multi record view with all records or record segments mapped as single, variable length character fields with field name "UnMapped" or "UnMappedSeg". No data conversion is performed.
`HEX`	Same as CHARACTER with the addition that the data is also displayed in Hex below the character display. Note that the Hex display occupies an additional 2 lines of data.

The data display format may be later altered using the FORMAT and/or ZOOM commands.

`READONLY`

Perform read-only edit of the data.

Unless the fileid of the edited data set is changed, attempts to save changes made to the data during the SDE edit session will give an error.

`REUSE`

Where possible, full edit capabilities of record move, copy, insert, change and delete are to be permitted for the data set being edited. Also, where the data set organisation and SDO structure permits, data changes may result in changes to record lengths.

SELCOPY/i SDE first determines whether the entire data set may be comfortably loaded into available storage. If not, parameter REUSE is ignored and AUXILIARY edit becomes default.

Full (REUSE) edit is not possible for VSAM RRDS and ESDS data sets defined with NOREUSE, in which case parameter REUSE is ignored and the ESDS/RRDS data set is edited using Update-in-place edit. For all other data sets, parameter REUSE is the default.

`UPDATE`

Update-in-place Edit is to be used so that existing records may only be altered and replaced without changing the record length. Records cannot be moved, copied, inserted or deleted.

Use of Update-in-place Edit has significant performance advantages over full (REUSE) edit when editing large, non-KSDS data sets.

UPDATE is default if the edited data set is a VSAM ESDS or RRDS file defined with NOREUSE or the data set is non-KSDS and its load into storage has been interrupted by the user via the Load Threshold break-in modal window.

`AUXILIARY`

Select Auxiliary edit for non-KSDS data sets. AUXILIARY is ignored for edit of a KSDS data set.

Auxiliary edit is identical to full (REUSE) edit except that the data set to be edited is first copied to an auxiliary data set on disk. This allows full editing capabilities for data sets that are too large to be loaded into available storage.

`FROM`

Records are to be displayed beginning at a specific location within the data set. The first record at the specified location becomes record 1 within the SDE window view of the data set. Records that occur before this location are not included within the edit session.

If FILTER is specified, then record filtering occurs only on records selected using the FROM and FOR parameters.

Required record location is identified via one of the following:

`KEY string`	For KSDS data sets only, locate the record with a key string equal to *string*. If not found, the record with the next key string greater than *string* is used.

`RBA n`	For KSDS and ESDS data sets only, locate the record starting at the relative byte address specified by *n*. The RBA must point at the start of the record otherwise a VSAM point error will occur. RBA address for each record is displayed as part of the record information columns within an SDE window view. Display of these columns is controlled using the SDE SET RECINFO CLI command.
`< RECORD > n`	For any data set organisation, locate the record number specified by *n*. Specification of parameter keyword RECORD is optional.

Use of the FROM parameter will force Update-in-place Edit.

Default is to display records within the SDE window view starting at the first record in the data set.

`FOR n_recs < RECORDS >`
Specifies the number of data set records to be included within the SDE window edit session. Specification of parameter keyword RECORDS is optional.

If FILTER is specified, then record filtering occurs only on records selected using the FROM and FOR parameters.

Use of the FOR parameter will force Update-in-place Edit.

Default is to include all records.

`FILTER filter_fileid | Filter Clause`
FILTER specifies additional record filtering criteria to further reduce the display of records for edit. All record filters are applied only to records that have been selected using the FROM and/or FOR parameters, otherwise it applies to all records in the structured data file.

Enough available storage must exist to load all records selected by the filter otherwise a storage error occurs and the edit is terminated.

FILTER parameters are specified via a filter clause which may be supplied as part of the EDIT command or referenced via *filter_fileid*, a separate sequential data set, PDS/PDSE member or HFS file. *filter_fileid* must contain the keyword FILTER followed by a valid filter clause.

The filter is applied only once during initial load of the data. A record will not be excluded from the display of edited records if its record type or data is changed in a way that no longer satisfies the filter criteria.

Use of FILTER will force Update-in-place Edit.

Filter Clause
A filter clause must be specified in "( )" (parentheses) and may contain comment data enclosed by "/*" and "*/". If filter clause is specified via *filter_fileid*, then comment data may also occur before and after the filter clause.

The following options are supported by the filter clause.

`INCLUDE record_type`
Include only records that are assigned the specified record type *record_type* by SDE Edit. This parameter may be specified repeatedly to include a number of record types or to perform alternative WHERE *expr* filters for the same record type. If INCLUDE is specified, then all record types that are not referenced by an INCLUDE parameter will be excluded by default.

record_type "Record" (with field name "UnMapped") may be used to perform a filter on the unformatted record data whether or not a structure (USING *struct_name*) has been specified. In this way, a filter may test **all** records regardless of their assigned record type.

INCLUDE and EXCLUDE parameters are mutually exclusive.

`EXCLUDE record_type`
Exclude only records that are assigned the specified record type *record_type* by SDE Edit. This parameter may be specified repeatedly to exclude a number of record types or to perform alternative WHERE *expr* filters for the same record type. If EXCLUDE is specified, then all record types that are not referenced by an EXCLUDE parameter will be included by default.

record_type "Record" (with field name "UnMapped") may be used to perform a filter on the unformatted record data whether or not a structure (USING *struct_name*) has been specified. In this way, a filter may test **all** records regardless of their assigned record type.

INCLUDE and EXCLUDE parameters are mutually exclusive.

`WHERE expr`
WHERE applies further filter conditions to records assigned to the record type specified by the last INCLUDE *record_type* or EXCLUDE *record_type* parameter processed.

expr is a valid SDE expression which supports function calls, *record_type* field names and references, sub-expressions, arithmetic, relational and logical operators. The result of the WHERE expression must be numeric and is treated as being Boolean in nature with a zero value indicating a "false" condition and any non-zero value indicating a "true" condition.

The WHERE expression is applied to each record assigned the record type *record_type* and, if the result is "true", the record is selected for include or exclude as indicated by the prevailing INCLUDE or EXCLUDE filter. If multiple INCLUDE/EXCLUDE *record_type* WHERE expressions exist for the same record type, then a logical OR is implied for all the expressions relating to that record type.

`STOPAFTER n_hits`
When the number of records selected by the INCLUDE or EXCLUDE filter reaches the value specified by STOPAFTER *n_hits*, then no further filter testing occurs.

If an INCLUDE filter, then all remaining untested records are excluded. If an EXCLUDE filter, then all remaining untested records are included.

`DB2 < (ssn) >`
Indicates that edit is for a DB2 base or results table. *ssn* is optional and identifies the local DB2 sub-system name to which a connection will be made.

Before a connection can be made to the DB2 sub-system, the SELCOPY/i DB2 plan must have been bound to that sub-system.

Default for *ssn* is the users default DB2 sub-system name as set by the DB2 Primary Options menu.

table_name
A DB2 base table name, specified as *table_owner.table_name*, containing rows of data to be edited.

view_name
A DB2 table view, specified as *view_owner.view_name*, which references an SQL query used to generate a results table containing the rows of data to be edited.

DB2 Opts
The following DB2 options are applicable to edit of DB2 table data only. See SDE Opts for structured data edit options.

`SELECT (select-clause)`
Specifies a DB2 SQL SELECT clause to be included in the prepared SQL select statement. See "*DB2 SQL Reference*" for syntax of the *select-clause*.

`WHERE (where-clause)`
Specifies a DB2 SQL WHERE clause to be included in the prepared SQL select statement. See "*DB2 SQL Reference*" for syntax of the *where-clause*.

`SORT|ORDER BY (order-by-clause)`

Specifies a DB2 SQL ORDER BY clause to be included in the prepared SQL select statement. See "*DB2 SQL Reference*" for syntax of the *order-by-clause*.

`USING (STRUCTURE) struct_name`
Identifies the SDO to be applied to the DB2 data rows. *struct_name* is the SDF fileid assigned to the SDO in a CREATE STRUCTURE command.

If the USING STRUCTURE option is specified then SELECT, WHERE and ORDER BY options are ignored. The SDO may contain DB2 SQL query SELECT, WHERE and ORDER BY sub-clause options which are used to select and display DB2 table rows. Other DB2 table edit options, apart from LOCKTABLE, FORMAT, READONLY, FROM and FOR, may also exist in the SDO and so define default values which may be overridden by specific parameters on the EDIT command. e.g. WITH isolation-clause, SKIPLOCKED, COMMIT.

`SKIPLOCKED`
Ignored unless an isolation level of Cursor Stability (CS) or Read Stability (RS) is in effect, SKIPLOCKED specifies that any selected rows that are already locked by another process should be skipped and not be included in the edit display. See "*DB2 SQL Reference*" for details on the SKIP LOCKED DATA clause.
Default is to allow display of locked rows whenever possible.

`PROTECTPRIMEKEY | EDITPRIMEKEY`
Specifies whether data occupying columns that comprise the table's primary key is eligible for update (EDITPRIMEKEY) or is read only (PROTECTPRIMEKEY).
Default is PROTECTPRIMEKEY.

`WITH CS|UR|RR|RS  < KEEP  EXCLUSIVE|UPDATE|SHR >`
Specifies a DB2 SQL isolation-clause to be included in the prepared SQL select statement. See "*DB2 SQL Reference*" for details on the *isolation-clause* and "*DB2 Performance Monitoring and Tuning Guide*" for details on the effects of isolation level on concurrency and protection of DB2 table data.

The KEEP sub-parameter options are applicable to isolation levels RR (Repeatable Read) and RS (Read Stability) only.
Default isolation level (as set by the SELCOPY/i DB2 package and plan BIND) is CS.

`COMMITONCLEANSAVE | COMMITONSAVE | COMMITONEXIT`
Specifies when a COMMIT should be executed for changed data.

`COMMITONCLEANSAVE`	COMMIT only if SAVE is executed without errors.
`COMMITONSAVE`	COMMIT on SAVE regardless of errors.
`COMMITONEXIT`	COMMIT only on exit of the edit session.

Default is COMMITONCLEANSAVE.

`LOCKTABLE EXCLUSIVE|SHR`
For DB2 base tables only, executes a DB2 SQL command LOCK TABLE before loading data from the table. Use this option with **caution** as other users and applications may be prevented from accessing the table. If LOCKTABLE is specified the lock is held until the next COMMIT is actioned.

LOCKTABLE EXCLUSIVE prevents another process from performing any operations on the table whilst it is being edited, unless the process is running with an isolation level of Uncommitted Read (UR) in which case read-only (dirty read) operations may be performed.

LOCKTABLE SHR prevents anything other than read-only operations to be performed on the table whilst it is being edited.

See "*DB2 SQL Reference*" for details on the effects of LOCK TABLE options.
Default is not to perform any explict table locking prior to load. (Recommended)

`COMMITONLOAD`
Perform a COMMIT following the initial load of rows to be edited, thus releasing all DB2 table locks performed during load of the data. This includes any explicit table locks applied via the LOCKTABLE parameter.
Default is not to perform a COMMIT following load of the table rows.

`AUDIT`
Open a new SELCOPY/i DB2 audit data set to record changes to the edited table made during this edit session. See Audit Trail Functions for details.
Default is not to perform edit auditing.

`FORMAT`
Specifies the format (TABLE or SINGLE) in which record data will be displayed in the initial SDE view. See FORMAT for structured data edit for details.
Default is FORMAT TABLE.

`READONLY`
Perform read-only edit of the data. Attempts to save changes made to the data during the SDE edit session will give an error.

`FROM < ROW > n`
Equivalent to FROM RECORD *n*, FROM ROW *n* specifies that rows are to be displayed beginning at row number *n* in the DB2 results table returned by the SQL query. The first row displayed becomes row 1 within the SDE window edit view. Rows that occur before this row number are not included within the edit session.
Default is row 1.

`FOR n_rows < ROWS >`
Equivalent to FOR *n_recs* RECORDS, FOR *n_rows* ROWS specifies the maximum number of rows to be displayed from the DB2 results table returned by the SQL query. Rows that fall outside the range of rows specified by FROM *n* FOR *n_rows* are not included within the edit session.
Default is to display all selected rows.

`PROFILE macro_name`
Specifies the REXX SDE edit macro to be executed as the profile when the data set is edited.

macro_name must exist in a library within the SDE macro path.

The PROFILE option only alters the profile used for the file currently being edited. It does not define the profile macro to be used for subsequent SDE edit.

The default is to execute a macro with member name SDEPROF if it exists.

`NOPROFILE`
Suppresses use of a profile macro when editing the file.

The NOPROFILE option only suppresses use of a profile for the file currently being edited. It does not suppress use of a profile macro for subsequent SDE edit.

Examples:

```
<sd edit CBL.SDE.MOD.ZJ2202  using CBL.CBLI.SDO(PRODL)  view ARCX1,PRDMST,PRODX
```
Edit data set "CBL.SDE.MOD.ZJ2202" using Structure Definintion Object (SDO) "CBL.CBLI.SDO(PRODL)" to map records.
Initially, all records of type ARCX1, PRDMST and PRODX are displayed.

```
<sd edit  CBL.SAMP01.KSDS  using CBL.CBLI.SDO(SAMP01)  from key XMHW003 for 50
```
Edit VSAM KSDS data set "CBL.SAMP01.KSDS" using SDO "CBL.CBLI.SDO(SAMP01)" to map 50 records starting at the record with key "XMHW003" (or the next record in key sequence if this key is not found.)

```
<sd edit  db2(CBLA)  ZZS.ZZSSYSMOD  where(SYSMOD LIKE 'RS%')  editprimekey
```
Edit filtered rows of DB2 table ZZS.ZZSSYSMOD in subsystem CBLA allowing update of the table's primary key columns.

See Also:

BROWSE
CREATE STRUCTURE
EDITDIALOG

EDITDIALOG

Syntax:

```
>>-- EDITDialog ------------------------------------------------------------><
```

Description:

Open the Structured Edit dialog window to edit or browse a data set using an SDE structure (SDO).

This dialog window may also be opened by selecting **Structured Edit** from the File menu in the CBLe frame window menu bar.

Parameters:

None

EMSG

Description:

SDE CLI command, EMSG, performs the same operation as the CBLe CLI command EMSG. See EMSG in CBLe Text Edit documentation.

END

Syntax:

```
>>--+-- END ---+-------------------------------------------------------------><
    |           |
    +-- QUIT --+
```

Description:

Close an SDE window view opened for EDIT or BROWSE of data. Set on **PF3** by default.

If additional SDE views exist for the data, then these will remain open.

For EDIT only, if only one SDE view exists for the data and that data includes unsaved changes, then one of the following will occur depending on the status of the AUTOSAVE option:

AUTOSAVE	**Action Taken**
ON (PROMPT or NOPROMPT)	Changes to the data are saved without prompting the user before doing so.
OFF NOPROMPT	Changes to the data are discarded. (Same as QQUIT)
OFF PROMPT	User is prompted to save the changes, discard the changes or to terminate the END command and return to the SDE edit view.

Furthermore, if a non-temporary structure (SDO) is used to map the record data and changes have been made to that structure during the course of the edit session (e.g. USE WHEN), then the user will be prompted to save the changed structure to its structured data file (SDF) regardless of AUTOSAVE status. See command, SAVESTRUCTURE.

Parameters:

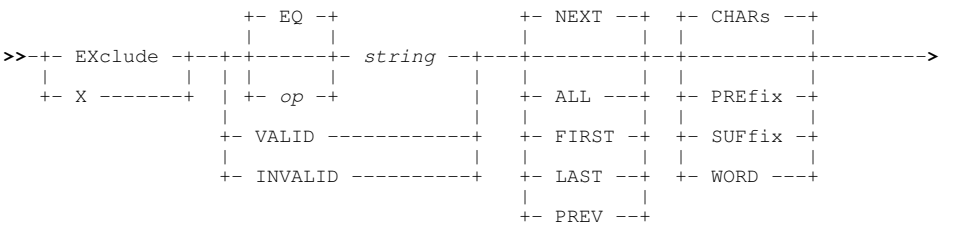
END has no parameters.

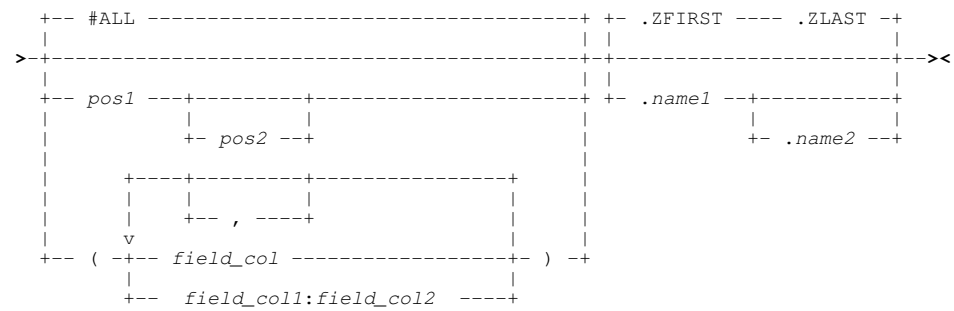
See Also:

SET AUTOSAVE
CANCEL
DROP, DDROP
QQUIT

EXCLUDE

Syntax:





Description:

Exclude data records in the current edit or browse view that satisfy a search for the specified character string or numeric value. EXCLUDE is not supported in Single Record View.

Only data records that are of the default record type are included in the search. Record groups that are already excluded will remain excluded, therefore, execution of successive EXCLUDE commands has a cumulative effect. Records groups that are of a different record type or are NOTSELECTED or SUPPRESSED, are excluded from the search.

EXCLUDE employs the same string search and field highlighting methods as the FIND command. All occurrences of the search string or numeric value that are not excluded by the EXCLUDE command, are highlighted in the text. A record group which has been excluded via the EXCLUDE command, will contain the highlighted field matching the search string if it is subsequently redisplayed. (e.g. using RESET EXCLUDED) Enter the RESET FIND command to turn off the highlighting.

Use of EXCLUDE with no parameters repeats the last EXCLUDE command executed including all its specified parameters.

The FORMAT of the SDE display affects the execution of EXCLUDE.

FORMAT CHAR and HEX:

A character compare for the supplied search string is performed against entire, unformatted data records.

field_col and #ALL parameters are not applicable to these display formats and are ignored.

FORMAT TABLE:

The search string is compared against **individual fields** that have been selected for display in the expanded data record. (See SELECT)

Fields are searched from left to right in the order that they appear in the display. This is true regardless of the order in which field columns are specified on the EXCLUDE command, or the order in which fields are encountered within the expanded record.

If a field column is specified on the EXCLUDE command that is not part of the display (e.g. an expanded group field or a field that has been removed from display by a SELECT command), then an error message is returned.

For character data type fields, a string compare is performed. For numeric data type fields (binary, packed decimal, floating point, zoned, etc.), then the following will occur:

1. If *pos1*, *pos2* positional parameters are **not** specified, the search string is interpreted as a signed numeric value and an arithmetic compare for equality is performed against the field's formatted numeric value.

   The length and data type of the numeric field, and the number of digits in the search value are not significant. e.g.

   ```
   EXCLUDE  67
   ```

   Excludes records containing a numeric field with value "67" (e.g. "0067", "67.00", "0.0670E+03") or a character field containing the string "67" (e.g. "167 Baker Street").

   If the search string is non-numeric, then numeric data type fields are not searched. Therefore, to bypass searching numeric data type fields, explicitly set the search string to be character or hex using 'string', C'string' or X'string' formats respectively. e.g.

   ```
   EXCLUDE    '67'
   EXCLUDE   C'67'
   EXCLUDE   X'F6F7'
   ```

   Excludes records only if a character field contains the string "67".

2. If *pos1*, *pos2* positional parameters are specified, the search string is interpreted as being a character string and a string compare is performed on the character representation of numeric data.

   Although the range of positions may span a number of fields, the search is still performed against individual fields within the range. i.e. a match for the search string will not be found if the data spans a field boundary.

   A match for the search string on just part of the character representation of a numeric field is considered to be a successful hit and the record is excluded. e.g.

   ```
   EXCLUDE    476    21  100
   ```

   Exclude records which contain a character or numeric field where character representation of the data contains the string "476". (e.g. a zoned decimal field with value "14760" or "-4762")

Parameters:

op

A relational operator used in the compare operation which determines the relationship that the data must have with the EXCLUDE search string in order for it to be identified as a successful match.

Valid values for *op* are as follow:

Operator	Description
EQ	Data must be equal to *string*. (Default)
NE	Data must be not equal to *string*.
GT	Data must be greater than *string*.
GE	Data must be greater than or equal to *string*.
LT	Data must be less than *string*.
LE	Data must be less than or equal to *string*.

If a character string compare is performed, the EBCDIC values assigned to characters in the search and data strings determine the relationship (equal to, greater than or less than) between the two strings.

string

The EXCLUDE search string. The seach string may be any of the following:

- ◊ An unquoted numeric value. The search string is treated as a numeric value when a numeric field is searched in a SINGLE or TABLE format view. In all other cases a numeric search string is treated as a character string.
- ◊ An unquoted character string containing no commas or blanks. The search for the character string will be case-insensitive so that uppercase and lowercase characters are treated as being the same.
- ◊ A character string enclosed in single (') or double (") quotation marks. The search string may contain embedded commas and blanks and the character string will be case-insensitive.

  Two adjacent quotation mark characters that are embedded in a search string which is enclosed by the same quotation mark characters, will be treated as a single occurrence of the character. e.g.

  ```
  EXCLUDE  'Jim O''Brien'
  ```

  Find the character string "Jim O'Brien".

◊ A character string enclosed in single (') or double (") quotation marks with the prefix C. This is equivalent to specifying a quoted search string but that the string search will be case-sensitive. (e.g. C'Book')

◊ A hexadecimal string enclosed in single (') or double (") quotation marks with the prefix X.

◊ A picture string enclosed in single (') or double (") quotation marks with the prefix P.

Picture strings use special characters to represent a generic group of characters as described below. Any character in a picture string that is not one of these special characters is untranslated.

String	Description
P'='	Any character.
P'¬'	Any non-blank character.
P'.'	Any non-displayable character.
P'#'	Any numeric character, 0-9.
P'-'	Any non-numeric character.
P'@'	Any uppercase or lowercase alpha character.
P'<'	Any lowercase alpha character.
P'>'	Any uppercase alpha character.
P'$'	Any non-alphanumeric special character.

If a no search string is specified, the default search string used is that specified on the most recent FIND, CHANGE, or EXCLUDE command executed against the current file. **EXCLUDE ALL** with no search string is a special case and will exclude all data records of the default record type.

`VALID`

Intended for use with formatted records, VALID will search fields for valid data. i.e. data that satisfies the field's assigned data type.

`INVALID`

Intended for use with formatted records, INVALID will search fields for invalid data. i.e. data that does not satisfy the field's assigned data type.

`ALL`

Search forwards from the top of the file data (i.e. the first position of the first data record) and excluded all records containing an occurrence of the string. EXCLUDE ALL with no other parameters excludes all records of the default record type.

`FIRST`

Search forwards from the top of the file data (i.e. the first position of the first data record) to exclude the first record of the default record type to contain an occurrence of the search string.

`LAST`

Search backwards from the bottom of the file data (i.e. the last position of the last data record) to exclude the last record of the default record type to contain an occurrence of the search string.

`NEXT`

Search forwards from the current cursor location to exclude the next record of the default record type to contain an occurrence of the search string. If the cursor is not within the window's data display area, the search begins at the first position of the first visible or excluded record within the display area that is of the default record type.

`PREV`

Search backwards from the current cursor location to exclude the previous record of the default record type to contain an occurrence of the search string. If the cursor is not within the window's data display area, the backwards search begins at the first position of the first visible or excluded record within the display area that is of the default record type.

`CHARS`

For non-numeric search strings only, CHARS indicates that a successful match occurs if the search string is found anywhere within the data being searched.

`PREFIX`

For non-numeric search strings only, PREFIX indicates that a successful match only occurs if the search string is found at the start of a word within the data being searched. i.e. the matched text must precede an alphanumeric character and either be preceded by a non-alphanumeric character or be at the start of a line or field.

`SUFFIX`

For non-numeric search strings only, SUFFIX indicates that a successful match only occurs if the search string is found at the end of a word within the data being searched. i.e. the matched text must be preceded by an alphanumeric character and either precede a non-alphanumeric character or be at the end of a line or field.

`WORD`

For non-numeric search strings only, WORD indicates that a successful match only occurs if the search string is found to be a complete word within the data being searched. i.e. the matched text must either be preceded by a non-alphanumeric character or be at the start of a line or field, and either precede a non-alphanumeric character or be at the end of a line or field.

`pos1`

The first position of a range of positions within the data record to be searched.

For TABLE format, this is a position in the expanded record. Only those fields, or parts of fields, that fall within the position range will be searched. Fields will be searched in the order that they occur within the display area.

pos1 may be a positive or negative integer value (not zero) and must be a value that is less than or equal to the maximum length of the data records or, for formatted record data, the length of the expanded record.

A negative value represents a position in the record relative to the end of the record. Therefore, where position 1 references the 1st character in the record, position -1 references the last character.

For all display formats, the string is treated as being non-numeric and the search is actioned on the character representation of the record data.

pos2
The last position of a range of positions within the data record to be searched.

Like *pos1*, *pos2* may be a positive or negative integer value (not zero). If *pos1* references a position within the record data which is higher than that referenced by *pos2*, then the *pos1* and *pos2* values are swapped.

If *pos2* is greater than the maximum length of the data records or, for formatted record data, greater than the length of the expanded record, then *pos2* is set equal to the maximum (or expanded) record length.

Default is *pos1* plus the length of the search string minus 1.

#ALL
Search all field columns in the current TABLE format display.

field_col
An individual field column to be searched within a TABLE format display.

The field column may be identified by its reference number (e.g. #6) or its name (e.g. JobID). If a field name is used, enclosing parentheses are **mandatory** Field names are automatically converted to a field reference and Referencing the same field column more than once will not cause an error.

If the field is an array, then all elements of the array are searched. To search an individual element of an array, the element occurrence should be specified in parentheses as a subscript to the field reference/name. e.g. #13(3) for the 3rd element of a single dimension array. An entry must exist for each dimension of the array. e.g. RoomSize(6,2,4) - a three dimensional array.

Specification of multiple field columns must either be enclosed in parentheses and/or separated by commas. The field columns may be specified in any order, however, the data is always searched from the first column in the display to the last.

Field column search specifications may be a combination of individual field columns and columns ranges. e.g. (JobID #6 Tax_Reference #12 #15:#20).

A search is only performed on field columns that are selected for display. Therefore, any field column specified on the EXCLUDE command that has not been selected for display, will be ignored.

field_col1:*field_col2*
The first and last fields of a range of field columns to be searched within a TABLE format display.

field_col1 and *field_col2* must be separated by ":" (colon) and if *field_col1* occurs after *field_col2* in the data record, then the values are swapped. *field_col1* and *field_col2* have the same specifications as *field_col*.

Specification of multiple field column ranges must either be enclosed in parentheses and/or separated by commas. Field column search specifications may be a combination of individual field columns and field columns ranges. e.g. (FirstName:LastName, #2, Salary:Bonus, EmpNo, #10:#15). If field names are used, enclosing parentheses are **mandatory**.

.name1
A label name identifying the first record of a range of data records to be searched. The preceding "." (dot) is mandatory. Default is .ZFIRST.

.name2
A label name identifying the last record of a range of data records to be searched. The preceding "." (dot) is mandatory. If *.name2* occurs before *.name1* in the display, then the order is reversed.
If EXCLUDE PREV is executed and *.name1* is specified, the default is .ZFIRST. Otherwise the default is .ZLAST.

Examples:

```
exclude last 100
```
Exclude the last record in the display (of the default record type) that contains a numeric field with value 100 or a character field containing the string "100".

```
ex c'Spec' prefix  (#10:#20, TitleTrack)
```
Exclude the first record in the display (of the default record type) that contains the exact string "Spec" as a word prefix within any of the selected character fields.

```
x all
```
Exclude all records of the default record type.

See Also:

CHANGE
FIND
RFIND
SELECT

EXTRACT

Description:

See SET/QUERY/EXTRACT Options.

FILE

Syntax:

```
>>-- FILE ------------------------------------------------------------------><
```

Description:

Close **all** SDE window views for data displayed in the current SDE view. Any unsaved changes are saved.

If a non-temporary structure (SDO) is used to map the record data and changes have been made to that structure during the course of the edit session (e.g. USE WHEN), then the user will be prompted to save the changed structure to its structured data file (SDF). See command, SAVESTRUCTURE.

Parameters:

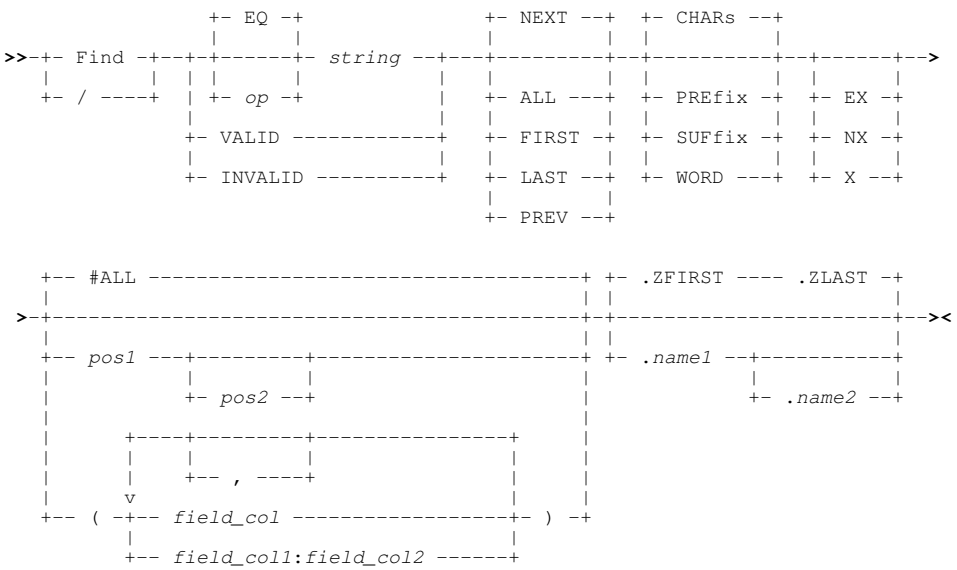
FILE has no parameters.

See Also:

CANCEL
END
QQUIT

FIND

Syntax:



Description:

Search data records in the current edit or browse view for the specified character string or numeric value. Only data records that are of the default record type (visible and EXCLUDED records) are included in the search. Records groups that are of a different record type or are NOTSELECTED or SUPPRESSED, are excluded from the search.

If the specified occurrence (all, first, last, next or previous) of the search string or numeric value is found within a record, then:

1. If the record is EXCLUDED, it is made visible.
2. The cursor is positioned at the beginning of the string or numeric field.
3. If necessary, scrolling occurs to display the found data.

All occurrences of the search string or numeric value are highlighted in the text. (Enter the RESET FIND command to turn off the highlighting.)

When the duration of a FIND or RFIND execution exceeds 1 second, a progress window is opened displaying the number of records processed so far. This window also provides the user with the oportunity to interrupt the operation using the Attn key.

Use of FIND with no parameters is equivalent to RFIND (assigned to function key **PF5** by default) and repeats the last find command executed, including all its specified parameters.

The FORMAT of the SDE display affects the execution of FIND.

FORMAT CHAR and HEX:

A character compare for the supplied search string is performed against entire, unformatted data records.

field_col and #ALL parameters are not applicable to these display formats and are ignored.

FORMAT SINGLE and TABLE:

The search string is compared against **individual fields** that have been selected for display in the expanded data record. (See SELECT)

Fields are searched from left to right in the order that they appear in the display. This is true regardless of the order in which field columns are specified on the FIND command, or the order in which fields are encountered within the expanded record.

If a field column is specified on the FIND command that is not part of the display (e.g. an expanded group field or a field that has been removed from display by a SELECT command), then an error message is returned.

For character data type fields, a string compare is performed. For numeric data type fields (binary, packed decimal, floating point, zoned, etc.), then the following will occur:

1. If *pos1*, *pos2* positional parameters are **not** specified, the search string is interpreted as a signed numeric value and an arithmetic compare for equality is performed against the field's formatted numeric value.

The length and data type of the numeric field, and the number of digits in the search value are not significant. e.g.

```
FIND  67
```

Finds numeric fields with value "67" (e.g. "0067", "67.00", "0.0670E+03") and character fields containing the string "67" (e.g. "167 Baker Street").

If the search string is non-numeric, then numeric data type fields are not searched. Therefore, to bypass searching numeric data type fields, explicitly set the search string to be character or hex using 'string', C'string' or X'string' formats respectively. e.g.

```
FIND   '67'
FIND  C'67'
FIND  X'F6F7'
```

Finds only character fields containing the string "67".

2. If *pos1*, *pos2* positional parameters are specified, the search string is interpreted as being a character string and a string compare is performed on the character representation of numeric data.

   Although the range of positions may span a number of fields, the search is still performed against individual fields within the range. i.e. a match for the search string will not be found if the data spans a field boundary.

   A match for the search string on just part of the character representation of a numeric field is considered to be a successful hit and the whole numeric field is highlighted. e.g.

   ```
   FIND   476    21  100
   ```

   Finds character and numeric fields where character representation of the data contains the string "476". (e.g. a zoned decimal field with value "14760" or "-4762")

Where a character string or numeric value is found in a numeric field, the entire field is highlighted and the cursor positioned at the start of the field.

Parameters:

op

A relational operator used in the compare operation which determines the relationship that the data must have with the search string in order for it to be identified as a successful match.

Valid values for *op* are as follow:

Operator	Description
EQ	Data must be equal to *string*. (Default)
NE	Data must be not equal to *string*.
GT	Data must be greater than *string*.
GE	Data must be greater than or equal to *string*.
LT	Data must be less than *string*.
LE	Data must be less than or equal to *string*.

If a character string compare is performed, the EBCDIC values assigned to characters in the search and data strings determine the relationship (equal to, greater than or less than) between the two strings.

string

The FIND search string. The search string may be any of the following:

◊ An unquoted numeric value. The search string is treated as a numeric value when a numeric field is searched in a SINGLE or TABLE format view. In all other cases a numeric search string is treated as a character string.

◊ An unquoted character string containing no commas or blanks. The search for the character string will be case-insensitive so that uppercase and lowercase characters are treated as being the same.

◊ A character string enclosed in single (') or double (") quotation marks. The search string may contain embedded commas and blanks and the character string will be case-insensitive.

Two adjacent quotation mark characters that are embedded in a search string which is enclosed by the same quotation mark characters, will be treated as a single occurrence of the character. e.g.

```
FIND  'Jim O''Brien'
```

Find the character string "Jim O'Brien".

◊ A character string enclosed in single (') or double (") quotation marks with the prefix C. This is equivalent to specifying a quoted search string but that the string search will be case-sensitive. (e.g. C'Book')

◊ A hexadecimal string enclosed in single (') or double (") quotation marks with the prefix X.

◊ A picture string enclosed in single (') or double (") quotation marks with the prefix P.

Picture strings use special characters to represent a generic group of characters as described below. Any character in a picture string that is not one of these special characters is untranslated.

String	Description
P'='	Any character.
P'¬'	Any non-blank character.
P'.'	Any non-displayable character.
P'#'	Any numeric character, 0-9.
P'-'	Any non-numeric character.
P'@'	Any uppercase or lowercase alpha character.
P'<'	Any lowercase alpha character.
P'>'	Any uppercase alpha character.
P'$'	Any non-alphanumeric special character.

If a no search string is specified, the default search string used is that specified on the most recent FIND, CHANGE, or EXCLUDE command executed against the current file. **FIND ALL** with no search string is a special case and is equivalent to executing RESET FIND EXCLUDED.

`VALID`

Intended for use with formatted records, VALID will search fields for valid data. i.e. data that satisfies the field's assigned data type.

`INVALID`

Intended for use with formatted records, INVALID will search fields for invalid data. i.e. data that does not satisfy the field's assigned data type.

`ALL`

If none of the records to be scanned are EXCLUDED or NX is specified, then FIND ALL is the same as FIND FIRST except that a message is displayed providing the number of occurrences of the string/value found in the data. Unlike FIND FIRST, **all** excluded records that contain an occurrence of the string/value are made visible if NX is not specified. FIND ALL with no other parameters is equivalent to RESET FIND EXCLUDED.

`FIRST`

Search forwards from the top of the file data (i.e. the first position of the first data record) to find the first occurrence of the string.

`LAST`

Search backwards from the bottom of the file data (i.e. the last position of the last data record) to find the last occurrence of the string.

`NEXT`

Search forwards from the current cursor location to find the next occurrence of the string. If the cursor is not within the window's data display area, the search begins at the first position of the first visible or excluded record within the display area that is of the default record type.

`PREV`

Search backwards from the current cursor location to find the previous occurrence of the string. If the cursor is not within the window's data display area, the backwards search begins at the first position of the first visible or excluded record within the display area that is of the default record type.

`CHARS`

For non-numeric search strings only, CHARS indicates that a successful match occurs if the search string is found anywhere within the data being searched.

`PREFIX`

For non-numeric search strings only, PREFIX indicates that a successful match only occurs if the search string is found at the start of a word within the data being searched. i.e. the matched text must precede an alphanumeric character and either be preceded by a non-alphanumeric character or be at the start of a line or field.

`SUFFIX`

For non-numeric search strings only, SUFFIX indicates that a successful match only occurs if the search string is found at the end of a word within the data being searched. i.e. the matched text must be preceded by an alphanumeric character and either precede a non-alphanumeric character or be at the end of a line or field.

`WORD`

For non-numeric search strings only, WORD indicates that a successful match only occurs if the search string is found to be a complete word within the data being searched. i.e. the matched text must either be preceded by a non-alphanumeric character or be at the start of a line or field, and either precede a non-alphanumeric character or be at the end of a line or field.

`EX`
`X`

Search EXCLUDED data records only.
FIND does not search records that are NOTSELECTED or SUPPRESSED.

`NX`

Search only visible data records (i.e. not EXCLUDED).
FIND does not search records that are NOTSELECTED or SUPPRESSED.

`pos1`

The first position of a range of positions within the data record to be searched.

For SINGLE and TABLE formats, this is a position in the expanded record. Only those fields, or parts of fields, that fall within the position range will be searched. Fields will be searched in the order that they occur within the display area.

pos1 may be a positive or negative integer value (not zero) and must be a value that is less than or equal to the maximum length of the data records or, for formatted record data, the length of the expanded record.

A negative value represents a position in the record relative to the end of the record. Therefore, where position 1 references the 1st character in the record, position -1 references the last character.

For all display formats, the string is treated as being non-numeric and the search is actioned on the character representation of the record data.

pos2
The last position of a range of positions within the data record to be searched.

Like *pos1*, *pos2* may be a positive or negative integer value (not zero). If *pos1* references a position within the record data which is higher than that referenced by *pos2*, then the *pos1* and *pos2* values are swapped.

If *pos2* is greater than the maximum length of the data records or, for formatted record data, greater than the length of the expanded record, then *pos2* is set equal to the maximum (or expanded) record length.

Default is *pos1* plus the length of the search string minus 1.

#ALL
Search all field columns in the current TABLE or SINGLE format display.

field_col
An individual field column to be searched within a TABLE or SINGLE format display.

The field column may be identified by its reference number (e.g. #6) or its name (e.g. JobID). If a field name is used, enclosing parentheses are **mandatory** Field names are automatically converted to a field reference and Referencing the same field column more than once will not cause an error.

If the field is an array, then all elements of the array are searched. To search an individual element of an array, the element occurrence should be specified in parentheses as a subscript to the field reference/name. e.g. #13(3) for the 3rd element of a single dimension array. An entry must exist for each dimension of the array. e.g. RoomSize(6,2,4) - a three dimensional array.

Specification of multiple field columns must either be enclosed in parentheses and/or separated by commas. The field columns may be specified in any order, however, the data is always searched from the first column in the display to the last.

Field column search specifications may be a combination of individual field columns and columns ranges. e.g. (JobID #6 Tax_Reference #12 #15:#20).

A search is only performed on field columns that are selected for display. Therefore any field column specified on the FIND command that has not been selected for display, will be ignored.

field_col1:*field_col2*
The first and last fields of a range of field columns to be searched within a TABLE or SINGLE format display.

field_col1 and *field_col2* must be separated by ":" (colon) and if *field_col1* occurs after *field_col2* in the data record, then the values are swapped. *field_col1* and *field_col2* have the same specifications as *field_col*.

Specification of multiple field column ranges must either be enclosed in parentheses and/or separated by commas. Field column search specifications may be a combination of individual field columns and field columns ranges. e.g. (FirstName:LastName, #2, Salary:Bonus, EmpNo, #10:#15). If field names are used, enclosing parentheses are **mandatory**.

.*name1*
A label name identifying the first record of a range of data records to be searched. The preceding "." (dot) is mandatory. Default is .ZFIRST.

.*name2*
A label name identifying the last record of a range of data records to be searched. The preceding "." (dot) is mandatory. If *.name2* occurs before *.name1* in the display, then the order is reversed.
If FIND PREV is executed and *.name1* is specified, the default is .ZFIRST. Otherwise the default is .ZLAST.

Examples:

```
find  22.3
```
Search all fields in the display belonging to the default record type for the next occurrence of the search string/value 22.3. Numeric fields are tested for numeric value 22.3, character fields are tested for character string "22.3" anywhere within the field.

```
find all 'ask' suffix  ex  (DsName:ProcLib, JobStep)
```
Search selected character fields in the display belonging to excluded records of the default record type for all occurrences of the word suffix string "ask".

See Also:

CHANGE
EXCLUDE
RFIND
SELECT

FLIP

Syntax:

```
>>-- FLIP ------------------------------------------------------------------><
```

Description:

Flip records that are of the default record type so that visible data records become EXCLUDED and EXCLUDED records are made visible.

Parameters:

FLIP has no parameters.

See Also:

LESS
MORE
MORE

FORMAT

Syntax:

```
>>-- FORmat --+-- Character ------+-------------------------------------------><
               |                  |
               +-- Hex ------------+
               |                  |
               +-- Single ---------+
               |                  |
               +-- Sngl -----------+
               |                  |
               +-- Tabl -----------+
```

Description:

Sets the display format for the current SDE BROWSE or EDIT window.

See also SDE command ZOOM to display unformatted data (FORMAT CHARACTER or FORMAT HEX) in single record display.

Parameters:

`Character`
: Multi record view with all records or record segments mapped as a single, character field with field name "UnMapped" or "UnMappedSeg". No data conversion is performed.

`Hex`
: Same as CHARACTER with the addition that the data is also displayed in Hex below the character display. Note that the Hex display occupies an additional 2 lines of data.

`Single`
`Sngl`
: Single record format with data types formatted according to the record structure. Each field occupies a separate line. Vertical scrolling transfers focus between fields. Horizontal scrolling transfers focus between records.

  By default, **PF2** is assigned to distributed CBLe edit macro SDEZOOMW which opens a new view of the record data and executes FORMAT SINGLE to display the record occupying the focus line in single format.

`Table`
: Multi-record table format (Default) with data types formatted according to the record structure. Each field occupies a separate column. Vertical scrolling transfers focus between records. Horizontal scrolling transfers focus between fields.

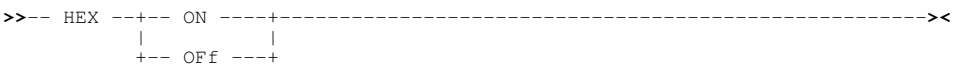
HELP

Description:

SDE CLI command, HELP, performs the same operation as the CBLe CLI command HELP. See HELP in CBLe Text Edit documentation.

HEX

Syntax:



Description:

Sets the hexadecimal display format on or off. Issued without the **ON** or **OFF** parameter the display format is toggled.

If the current display format has been set using **FORMAT CHAR**, or **FORMAT HEX**, then **HEX ON** is the equivalent of issuing **FORMAT HEX**.

If the current display format has been set using **FORMAT SINGLE**, or **FORMAT TABLE**, then for each line displayed **HEX ON** will add two further lines to show the hexadecimal representation of the raw (unformatted) data below each formatted field. e.g.

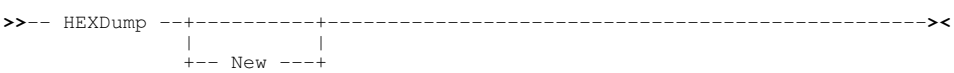
```
 SRCRecOff  SRCRecLen SRCMbr
       #44        #45 #47
  BN 121:4   BN 125:4 AN 137:8
<---+----> <---+----> <---+-->
       152         80 APEZLINK
      0009       0005 CDCEDCDD
      0008       0000 17593952

       144         80 SDEFCMD1
      0009       0005 ECCCCDCF
      0000       0000 24563441
```

Parameters:

`ON`
`OFF`
: HEX display format is set ON or OFF.

HEXDUMP

Syntax:



Description:

Display the focus record or record segment in single record, unformatted hex dump view. HEXDUMP may also be executed for an individual record/record segment using prefix command HEX or HEXD.

The hex dump view is a storage display window which displays a number of fullwords of data in both long hexadecimal and character representation. The hex dump display supports the storage window options popup menu, opened by positioning the cursor in the character or hex representation of the data and pressing <PF4> or executing primary command SPM (SHOWPOPUPMENU).

```
  Record type: REC-CARD   Fixed(66) Offset=0 Data elements=13

  Record> 00000008   Flags: f        Length:     66

       1 00000000 00000008 0000061C 0001C3C4         .  ... .CD
      17 C5D3E340 40400675 48756764 74656CD4 ELT    .ÍçÍÅÀÈÀ%M
      33 D940C340 E2D3C1C9 D5404040 40404040 R C SLAIN
      49 40404040 40404040 0200409C 0200501C          .  æ. &.
      65 719C                               Éæ
```

Unlike other storage display windows, altering the number of fullwords displayed per line and/or including display of the data's address in storage will not attempt to resize the window display.

For EDIT, both the hexadecimal and character representations of the data may be updated whereby changes to one representation of the data are automatically reflected in the other.

Beware, however, that overtyping the character representation of the data will update **all** fullwords in the updated line so they reflect the character data. In the character representation, a null character (x'00') is represented by blank (EBCDIC x'40') and other non-printable characters are represented by dots (EBCDIC x'4B'). Therefore, updating the character representation will result in these characters replacing the original hexadecimal values.

If parameter NEW is specified, the hex dump display is opened in a new window view allowing the user to preserve the current data display format in the original window view.

Parameters:

`NEW`
: Open the hex dump display in a new SDE EDIT/BROWSE window view.

See Also:

CHAR
MAP (FMT)
UNMAP (UNFMT)
VFMT
ZOOM

HIDE

Syntax:

```
>>-- HIDE ------------------------------------------------------------------><
```

Description:

Hide all shadow lines. HIDE is equivalent to SET SHADOW OFF ALL.

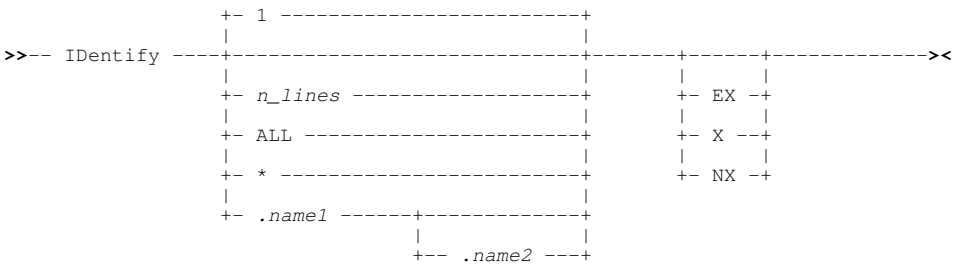
RESET HIDE will redisplay all shadow lines.

Parameters:

HIDE has no parameters.

IDENTIFY

Syntax:



Description:

IDENTIFY is used to remap records (re-assign record-type RTO) in the current SDE edit view. For segmented record edit, if any segment within a record is selected for remap by the IDENTIFY command, then all segments within the record will be remapped.

By default, IDENTIFY will not attempt to select and remap records that have **not** been altered since the file was loaded for edit.

See *"Record Type Assignment"* for description of the processing performed on each record/record segment selected by IDENTIFY for remap.

ID fields are identified by a record type (RTO) definition containing USE WHEN criteria. Values in these fields determine whether that RTO is suitable to be assigned to an individual record or record segment.

SELCOPY/i SDE is sensitive to changes made to ID fields that have been identified explicity by field name or field reference number in the USE WHEN expresssion. It is also sensitive to changes to data in when record-type formatting is disabled (i.e. field name "Unmapped" or field reference number #1).

Where one of these ID field changes is detected, or where the length of the record/segment data is altered (via CHANGE or RECLEN updates), the ID flag (**==ID>**) is set on for the changed record/segment. This is intended to notify the user that the record/segment data may no longer satisfy the USE WHEN criteria for its assigned record type.

In order to allow the user the opportunity to complete any changes being made, SELCOPY/i does not automatically remap records with the ID flag set on.

The records selected for remap by IDENTIFY depend on the current value of the IDSCOPE option. By default, IDSCOPE is set to CHANGED indicating that all records flagged as having been changed may be selected for remap. Alternatively, IDSCOPE FLAGGED indicates that only records with the ID flag set on may be selected for remap.

The remap of records, performed on execution of IDENTIFY, may be individually undone and subsequently redone using the UNDO (<PF22>) and REDO (<PF23>) commands.

Parameters:

Note that in single-record view the SDE window view displays the focus line (record or record segment) of the equivalent multi-record view.

n_lines
: Specifies the number of lines (records or segments), including the focus line, to be scanned searching for records/segments to be selected for remap.
The default is 1 line.

ALL | *
: Specifies that all lines in the file are to be scanned, searching for records/segments to be selected for remap.

.name1
: A label name identifying the first line of a range of lines to be scanned, searching for records/segments to be selected for remap. The preceding "." (dot) is mandatory.

.name2
: A label name identifying the last line of a range of lines to be scanned, searching for records/segments to be selected for remap. The preceding "." (dot) is mandatory.
If *.name2* occurs before *.name1* in the display, then the order is reversed.
If *.name1* is specified without *.name2*, then .ZLAST is the default.

EX | X
: Search for records/segments occupying EXCLUDED lines only.

NX
: Search for records/segments occupying visible (non-EXCLUDED) lines only.
Default is to search for records/segments in both EXCLUDED and non-EXCLUDED lines.

See Also:

SET/QUERY/EXTRACT IDSCOPE

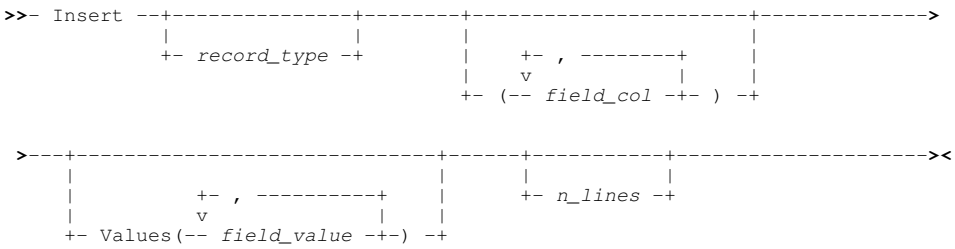
IMMEDIATE

Description:

SDE CLI command, IMMEDIATE, performs the same operation as the CBLe CLI command IMMEDIATE. See IMMEDIATE in CBLe Text Edit documentation.

INSERT

Syntax:



Description:

Insert one or more new records following the focus line, optionally providing explicit values to be inserted into specified fields.

Insert of new records is not allowed for Update-in-place editing unless the data set is an RRDS or VRDS. For these types of files, records that are empty slots may be activated using the INSERT command where, starting from the focus line, *n_lines* represents the number of lines to be searched for empty slots. All empty slots found within this range of lines will be activated.

Fields within the inserted records do not require explicitly assigned values. This occurs either when no list of field values is specified or when the default record type contains more fields than there are values in the specified list of field values.

If fields are not explicitly assigned values (either because no list of field values is specified or because the default record type contains more fields than there are values in the specified list of field values) then they are assigned default values as follow:

- 0 (zero) for numeric fields.
- Nulls for bit and hex fields.
- Blanks for all other field types.

If a USE WHEN expression has been defined for the chosed record type, then an attempt is made to populate any fields specified in that expression with values which satisfy the expression.

INSERT with no parameters inserts one record of the current default record type with default values in all fields.

If no structure (SDO) or copybook has been applied to data records of variable length (i.e. unformatted variable length records), then the user is prompted to specify a record length for the inserted record. This is because the SDE editor preserves record length when the records are saved. See Changing Record Lengths.

Following execution of INSERT, the DRECTYPE setting is updated to be the record type specified by the *record_type* parameter.

Also see the "I(n)" insert prefix area command.

Parameters:

record_type
: The record type of the record(s) to be inserted. If *record_type* is not specified, then the default record type is used.

field_col
: An individual field column within a list of field columns for which a corresponding *field_value* is to be inserted.

The field column may be identified by its reference number (e.g. #1) or its name (e.g. SeqNUM). Specification of multiple field columns must be separated by commas and enclosed in parentheses.

If the field is a **struct** or a **union**, then an error message is returned. If the field is an array, then an individual array element must be specified in parentheses as a subscript to the field. e.g. #13(3) for the 3rd element of a single dimension array. A subscript entry must exist for each dimension of the array. e.g. RoomSize(6,2,4) - an element within a three dimensional array.

If a list of field columns is not specified then as many of the set of selected fields in the current view as there are values in the list of field values, are used as the default. In this case, the order of the field columns is equal to the order in which fields are displayed in the current view.

`field_value`
An individual field value within a list of field values to be inserted in a corresponding *field_col* entry in the field column list.

If one or more *field_col* entries are specified (i.e. a field list is specified), then there must be the same number of *field_value* values to correspond with each *field_col* entry.

If a field value contains special characters, blanks or commas, then it should be enclosed in quotes.

`n_lines`
Number of identical records to be inserted or, for RRDS/VRDS Update-in-place edit, the number of lines to be searched for Empty Slots.
Default is 1.

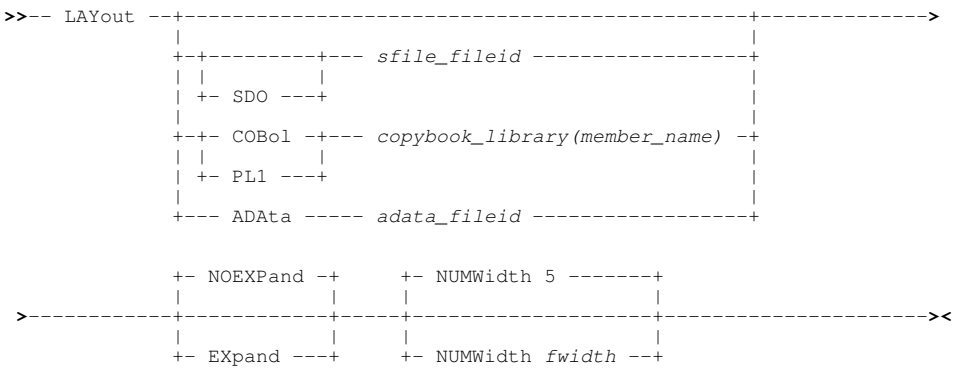
Examples:

`insert (SeqNum, FirstName, LastName, JobTitle, Salary) Val(283,Jacob,Smith,'Systems Engineer',35,950.00)`
Insert a record of the default record type populating the specified list of fields with the explicit values in the corresponding list of field values. All other fields are populated with default values.

`insert 2 Val(18.3,"Michael O'Leary",-12.333)`
Insert 2 records of the default record type populating the first 3 selected fields in the display with the explicit values specified in the list of field values. All other fields are populated with default values.

LAYOUT

Syntax:



Description:

Display the record structure layout list window detailing all record-types in the specified structure file (SDE structure (SDO), COBOL/PL1 copy book or COBOL/PL1 ADATA file).

If executed from an SDE browse/edit view without specifying the name of a structure file, the layout window is displayed for all record-types in the structure used to may records in the current display.

If executed from any other window with no parameters the Display Record Layout panel (=9.3) is opened.

Parameters:

`SDO sfile_fileid`
Identifies an existing SELCOPY/i SDE structure (SDO) file, *sfile_fileid*

`COBOL|PL1 copybook_library(member_name)`
Identifies a COBOL or PL1 copy book PDS/PDSE library and member name, *copybook_library(member_name)*.

`ADATA adata_fileid`
Identifies a COBOL or PL1 ADATA output file, *adata_fileid*.

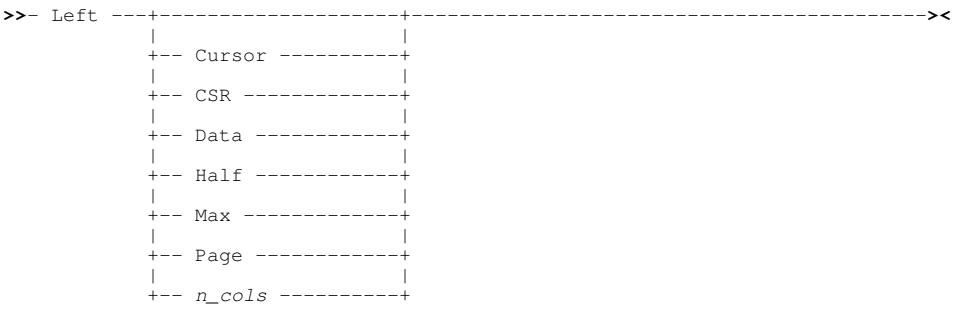
`NUMWIDTH fwidth`
Specifies the displayed width of numeric columns: **RefNo**, **Start**, **End** and **Length** in the layout list output.

The default (and minimum) width of these fields is 5 characters.

`EXPAND | NOEXPAND`
Specifies whether or not array (OCCURS) fields are to be expanded to display every repeating instance of a field within that array.
Default is NOEXPAND.

LEFT

Syntax:



Description:

In multi record view, scroll to the left the display of field and header lines belonging to records that are of the default record type. The display of all other visible records, header lines and shadow lines remains unchanged.

In single record view, LEFT will display the fields belonging to a visible data record that has a lower line number than the record (or record segment) that is in the current view. For display of segmented records, LEFT and RIGHT scroll through each segment in the file, regardless of the record to which it belongs. NEXT and PREV may be used to scroll between secondary segments of the current record only. LEFT CURSOR/DATA/HALF/PAGE are not applicable in single record view.

LEFT is assigned to **PF10** by default. Any characters specified on the command line when the PFKey is hit will be concatenated to the command and treated as a parameter string.
Where no parameter is specified, the scroll amount will be the value specified in the "**Scroll>**" field.

Multi Record View Scrolling:

Columns may have the HOLD flag set on by the SELECT command. These columns are static in the display and, like the prefix area and Lrecl column, are unaffected by LEFT and RIGHT scrolling. Columns that have the HOLD flag set off are referred to as **floating** columns.

The following interpretation of cursor location applies so that LEFT CURSOR can operate successfully on the appropriate data:

- Cursor is located at an offset within a column header line.
  The cursor position is interpreted as being at the same offset within the data record that immediately follows the group of header lines.
- Cursor is located within a column of type character (AN) but beyond the width of the character data.
  The cursor position is interpreted as being the last position of the displayed character data.
- Cursor is located immediately to the left of a column of any type.
  The cursor position is interpreted as being the the first position of the data displayed within that column.

Where the last column to be displayed is not of type character (AN), the magnitude by which the display is scrolled to the left is always reduced to display all field data belonging to the last column. In addition, the display area must contain all column header text belonging to the first column and, where the first column is not of type character (AN), it must also contain all field data belonging to the first column. Therefore, to accomodate this, the magnitude by which the display is scrolled to the left may be further reduced.

Because of these adjustments, it is possible that execution of a LEFT command may result in no change to the displayed data, in which case LEFT PAGE is executed instead.

No adjustment is made for scrolling left into an offset within a character data column. i.e. where the last column of the display is of type character, the last position of the column display need not be the last character of the character data. If the last column to be displayed is of type character and the display width is too small to accomodate all the column's data and header text, the column will be truncated. However, due to the other adjustments made for scrolling left, it may be impossible for some characters within the character data to occupy the last position of the column display. In order to overcome this, the user would have to alter the display area dimensions.

Attempting to scroll left beyond the start of the record will make the first data column the first column of the display.

Parameters:

`CURSOR`
`CSR`
The focus column becomes the last column of the scrolled display.
In addition to this, if the focus column is type character (AN) and the cursor is positioned on a character that is at an offset into the focus column data, then an attempt is made to make that character occupy the last position of the scrolled display.

If any of the following conditions are true, then LEFT PAGE is executed instead:

◊ Cursor is positioned anywhere other than within a floating column in a visible data record.
◊ Focus column is **not** type AN and is already the last column of the display.
◊ Focus column is type AN, is already the last column of the display and cursor position is at the last position of the displayed column data.

`DATA`
Scroll left so that the first floating column in the current display area becomes the last column of the scrolled display.
If this column is type character (AN) and the first position of the display falls on a character that is at an offset into the column, then an attempt is made to make that character occupy the last position of the scrolled display.

`HALF`
Scroll a number of columns so that the column situated half way along the width of the current display of floating columns, becomes the last column of the scrolled display.
If this column is type character (AN) and the half display position falls on a character that is at an offset into the column, then an attempt is made to make that character occupy the last position of the scrolled display.

`MAX`
Scroll left to display the first floating column of data in a multi record view.

In single record view, the first visible data record is displayed.

`PAGE`
Scroll left so that the floating column of data to the left of the first floating column in the current display, becomes the last column of the scrolled display.
If this column is type character (AN) and the first position of the display minus one falls on a character that is at an offset into the column, then an attempt is made to make that character occupy the last position of the scrolled display.

`n_cols`
In a multi record view, scroll left a specified number of floating columns.
The floating column of data that is *n_cols* to the left of the first floating column becomes the new first floating column of the scrolled display.

In single record view, the visible data record that is *n_cols* records before the record currently in view, gets displayed.

Examples:

`left page`
Scroll records of the default record type in a multi record view display area, left by a full display area width minus the width of any columns flagged with HOLD.

See Also:

DOWN
RIGHT
UP

LESS

Syntax:

```
>>--- LESS ---+-- where_clause --+-------------------------------------------><
              |                  |
              +-- line_flag -----+
```

Description:

Exclude any visible records that are of the default record type and also satisfy the specified *where_clause* criteria. Records that are already EXCLUDED are unaffected by the LESS command. The current line is also unaffected unless the SHADOW option for EXCLUDED lines is set off and the current line is one which is excluded by the LESS command. In this case the line following the current line becomes the new current line.

If LESS is executed with no parameters, then all visible records that are of the default record type are EXCLUDED.

The record type used on a LESS command becomes the new value of the DRECTYPE option.

Parameters:

where_clause
where_clause syntax is described fully in documentation for the WHERE command.

line_flag
line_flag may be specified to locate a record or record segment that has been flagged with the requested line flag.

Each valid *line_flag* keyword detailed below corresponds to a built-in function.

A description of each *line_flag* keyword may be found in its equivalent function description.

***line_flag* Keyword**	**Built-in Function**
ERRor	CHANGEERROR()
CHAnge | Chg	CHANGEOK()
ALTered | UPDated	CHANGED()
DATaerror	DATATYPEERROR()
DUPkey	DUPLICATEKEY()
EMPTY	EMPTYSLOT()
EXcluded | X	EXCLUDED()
LABel	HASPOINT()
COMmand | CMd	HASPREFIXCMD()
IDentify | IDrequired	IDREQUIRED()
NEW	INSERTED()
KEYChanged | KEYChg	KEYCHANGED()
LENgtherror	LENGTHERROR()
EOL | NOEOL	NOEOL()
SAVE	SAVEREQUIRED()
TRNC | TRUNCated	TRUNCATED()
VALERRor	VALUEERROR()

Examples:

```
less    #14 >= 255
```
Exclude any records that are of the default record type where numeric field reference number 14 is greater than or equal to 255.

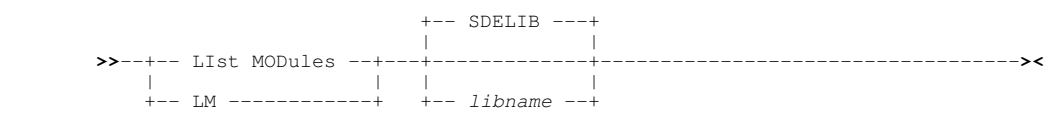
```
less    OrderDate <= PaymentReceivedDate
```
Exclude any records that are of the default record type where the contents of field "OrderDate" is less than or equal to the contents of field "PaymentReceivedDate".

See Also:

FLIP
MORE
WHERE

LIST MODULES

Syntax:



Description:

Use the LIST MODULES command to display a list window containing details of modules in the specified SELCOPY/i function library.

LIST MODULES may be executed to determine the level of any currently installed SELCOPY/i function module. CBL may request that this command is executed for diagnostic purposes in problem determination.

Parameters:

libname
The name of a SELCOPY/i library. Valid entries include VCILIB, WINLIB, EDTLIB, SDELIB, TABLIB, etc.

Default is SDELIB.

LIST RPOS

Syntax:

Description:

Use the LIST RPOS command to display a list window containing details of RPOs (Record Position Objects) for the current SDE edit/browse view.

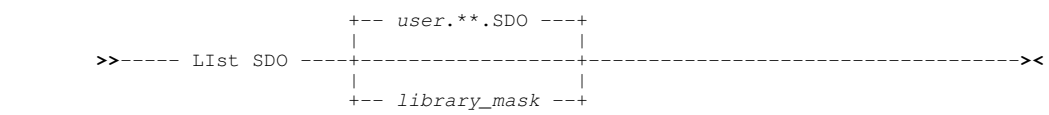
CBL may request that this command is executed for diagnostic purposes in problem determination.

Parameters:

LIST RPOS has no parameters.

LIST SDO

Syntax:



Description:

Use the LIST SDO command to display the Structure Definition Members list window containing details of each SDE structure (SDO) library member selected by the nominated PDS/PDSE library and member masks.

Parameters:

library_mask
Specifies a PDS/PDSE library DSN mask and optionally one or more member name masks. If no member mask is specified, all members in selected libraries will be listed.

A library DSN mask may contain the following wild card characters:

`*` A single asterisk represents a DSN qualifier or zero or more characters within a DSN qualifier.
e.g. `DEV.*.SDO`, `DEV.C*.*.SDO`

`**` Double asterisk represents zero or more qualifiers within a DSN. Double asterisk must be preceded or followed by either a "." (dot/period) or a blank. It cannot precede or follow an alphanumeric character.
e.g. `DEV.**.SDO`, `DEV.SDO.**`

`%` A single percent sign represents exactly one character other than "." (dot/period) within a DSN qualifier.
e.g. `DEV.C%%.SDO`

One or more member masks may be specified between a single pair of "( )" (parentheses). Multiple PDS/PDSE member masks must be separated by a "," (commma) and/or one or more intervening blanks.

A member mask may contain the following wild card characters:

`*` A single asterisk represents an entire member name or zero or more characters within a member name.

`%` A single percent sign represents exactly one character within a member name mask.

Default is library mask is "*user*.**.SDO" where *user* is the user's RACF logon id.

Examples:

`LIST SDO DEV.%BL*.**.SDO(CC*, *MAN, XM*J*)`
List SDO members that match this library and member masks.

LIST STORAGE

Syntax:

```
>>----- LIst STOrage ------------------------------------------------------><
```

Description:

Use the LIST STORAGE command to open a list window containing details of bit mapped storage used for the current SDE edit/browse view.

CBL may request that this command is executed for diagnostic purposes in problem determination.

Parameters:

LIST STORAGE has no parameters.

LIST UKRS

Syntax:

```
>>----- LIst UKRs ------------------------------------------------------><
```

Description:

For SDE edit of a VSAM KSDS data set, use the LIST UKRS command to open a list window containing details of updated KSDS keys for the current SDE edit view.

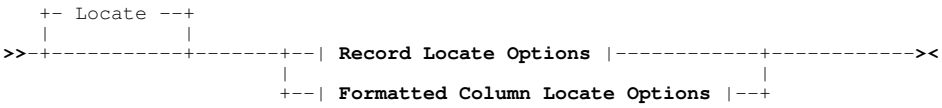
CBL may request that this command is executed for diagnostic purposes in problem determination.

Parameters:

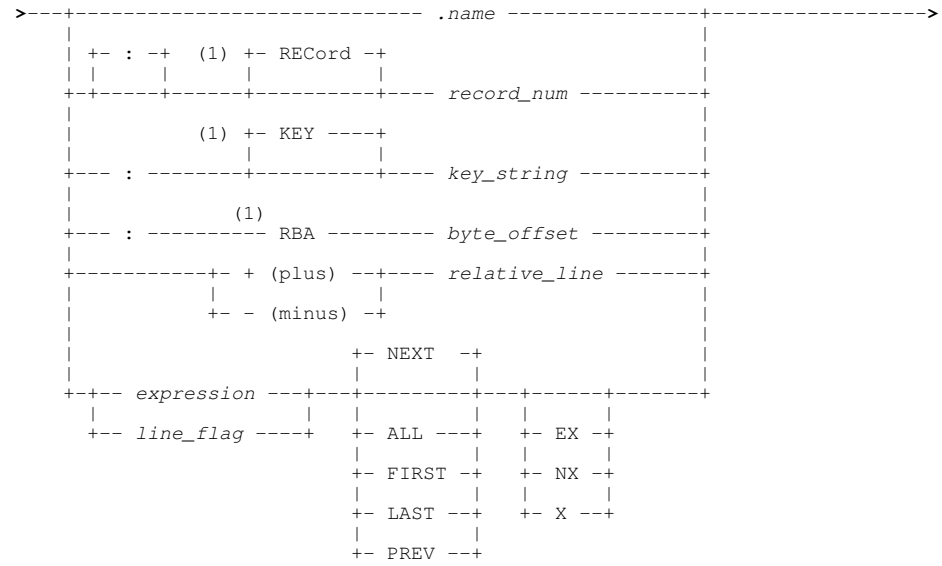
LIST UKRS has no parameters.

LOCATE

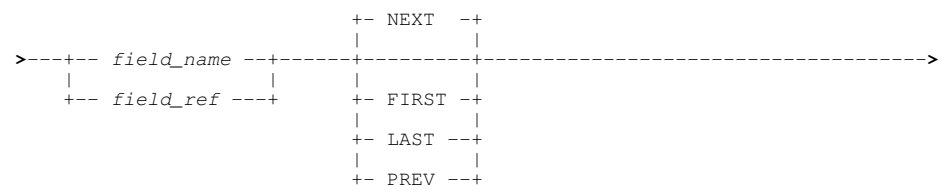
Syntax:



Record Locate Options:



Formatted Column Locate Options:



Notes:

1. Parameters RECORD, KEY or RBA may be specified before or after their associated *record_num*, *key_string* or *byte_offset* value.

Description:

Locate and scroll to a data record or the formatted record column field that matches the specified criteria. Where no match is found, the display is unchanged.

Field column location is supported for formatted record data (i.e mapped by an SDO structure record type) and applies to the default record. only.

In single record view, record locate will scroll left or right to display the record that matches the locate criteria, whereas formatted column locate will scroll up or down so that the specified field appears first in the display.

The LOCATE command verb may be omitted altogether if the first token in the command can be identified as being a valid locate parameter and ambiguity exists with defined SELCOPY/i command verbs.

If *expression* is specified for record location and no field name nor field reference is included within *expression*, then the LOCATE is performed only for records that are of the default record type. Records that are not of the default record type are bypassed.

When the duration of a LOCATE execution exceeds 1 second, a progress window is opened displaying the number of records processed so far. This window also provides the user with the opportunity to interrupt the operation using the Attn key.

Parameters:

Record Locate Options

`.name`
: Label name assigned to a line within the data display.
The preceding "." (dot) is mandatory.

`:`
: A ":" (colon) identifies the locate target to be an absolute record number, KSDS data set key value or VSAM RBA (relative byte address.)

Required target record is identified via one of the following:

`(RECORD) record_num`
: For any data set organisation, locate the record number specified by *record_num*.

For edit techniques where record numbers have not yet established (e.g. KSDS edit following a LOCATE by KEY, or DOWN MAX), then locating by record number will establish the record numbers and update the prefix area display accordingly.

Specification of parameter keyword RECORD and leading ":" (colon) is optional if *record_num* is an unquoted numeric value. e.g. `5`, `212`

`(KEY) key_string`
: For KSDS data sets only, locate the record with a key string equal to *key_string*. If not found, the record with the next key string greater than *key_string* is located.

key_string may be specified as:

1. An unquoted character string. The string must contain no commas or blanks and will be upper cased.
2. A character string enclosed in apostrophes (') or quotation (") marks. The string will be upper cased and any two adjacent characters embedded in the string that are the same as the enclosing symbols, are treated as a single occurrence of the character.
3. A character string enclosed in apostrophes (') or quotation (") marks with the prefix (or suffix) C. This has characteristics equivalent to key strings specified in the previous item except that no upper casing will occur.
4. A hexadecimal string enclosed in apostrophes (') or quotation (") marks with the prefix (or suffix) X.

Specification of parameter keyword KEY is optional if *key_string* is **not** an unquoted numeric value. e.g. `:ABC`, `:12UF00`, `:'123456'`, `:X'003D67A6'`, `:C'abc_00'`

`RBA byte_offset`
: For KSDS and ESDS data sets only, locate the record starting at or following the relative byte address specified by *byte_offset*.

By default, VSAM returns a point error if the specified RBA does not point at the start of a record. SDE detects this condition so that, if the RBA does not point at the start of a record, then the record with the next higher RBA becomes the current line of the display. For KSDS data sets, a best effort attempt will be made to locate the record with the next higher RBA though this may not be accurate.

RBA address for each record is displayed as part of the record information columns within an SDE window view. Display of these columns is controlled using the SDE SET RECINFO CLI command.

byte_offset may be specified as a decimal or hexadecimal numeric value. e.g. `:RBA X'01D5'` is equivalent to `:RBA 469`

Specification of parameter keyword RBA is mandatory.

Parameter keywords RECORD, KEY and RBA may be specified before or after their associated locate line target values. e.g. `:KEY C'APE110X5'` is equivalent to `:C'APE110X5' KEY`

If the target line contains a record that is flagged as being EXCLUDED, SUPPRESSED or NOTSELECTED, then the following message is returned:

```
ZZSD177E Requested line record_num|key_string|byte_offset is not visible.
```

`relative_line`
: An integer number representing the number of lines through which the display is to be scrolled. The lines scrolled include data records and shadow line groups.

relative_line may be preceded by "+" (plus) or "-" (minus) which specifies the direction, down or up respectively, in which the display will be scrolled. If omitted, "+" is assumed. (e.g. +18, -2, 3)

Specifying *relative_line* is equivalent to executing the DOWN *n_lines* or UP *n_lines* commands.

expression

expression is a valid SDE expression which supports function calls, *record_type* field names and references, sub-expressions, arithmetic, relational and logical operators. The result of *expression* must be numeric and is treated as being Boolean in nature with a zero value indicating a "false" condition and any non-zero value indicating a "true" condition.

When an expression is applied to the record, it must test "true" in order to be selected as a target of the LOCATE command.

For LOCATE, *expression* may not be specified as a single field value term with no operators. An expression of this sort will be treated as a formatted column locate specification. Furthermore, if *expression* involves a function call to a built-in function name that matches a subscripted field name in the default record, then the function call will be interpreted as a field value. To force the function call, the internal function name should be used instead (i.e. BIF_*function_name*).

By default, data records of the selected record type that have been EXCLUDED are included in the locate scan and are made visible if they satisfy *expression*.

line_flag

line_flag may be specified to locate a record or record segment that has been flagged with the requested line flag.

Each valid *line_flag* keyword detailed below corresponds to a built-in function. Because field column name location will take precedence, if the *line_flag* keyword conflicts with a *field_name* in the default record type, then the equivalent built-in function name should be used instead.

A description of each *line_flag* keyword may be found in its equivalent function description.

***line_flag* Keyword**	**Built-in Function**
ERRor	CHANGEERROR()
CHAnge | Chg	CHANGEOK()
ALTered | UPDated	CHANGED()
DATaerror	DATATYPEERROR()
DUPkey	DUPLICATEKEY()
EMPTY	EMPTYSLOT()
EXcluded | X	EXCLUDED()
LABel	HASPOINT()
COMmand | CMd	HASPREFIXCMD()
IDentify | IDrequired	IDREQUIRED()
NEW	INSERTED()
KEYChanged | KEYChg	KEYCHANGED()
LENgtherror	LENGTHERROR()
EOL | NOEOL	NOEOL()
SAVE	SAVEREQUIRED()
TRNC | TRUNCated	TRUNCATED()
VALERRor	VALUEERROR()

ALL

Locate all records that match the specified *expression* expression and position the display at the first matching record. All records for which *expression* test "true" are made visible if NX is not specified.

FIRST

Search forwards from the first record in the file with the selected record type to locate the first record that satisfies the specified *expression*.

LAST

Search backwards from the last record in the file with the selected record type to locate the last record that satisfies the specified *expression*.

NEXT

Search forwards from the focus line to locate the next record that satisfies the specified *expression*.

PREV

Search backwards from the focus line to locate the previous record that satisfies the specified *expression*.

EX
X

Locate EXCLUDED data records only.
LOCATE does not search records that are NOTSELECTED or SUPPRESSED.

NX

Locate only visible data records (i.e. not EXCLUDED).
LOCATE does not search records that are NOTSELECTED or SUPPRESSED.

Formatted Column Locate Options

Formatted column locate is applicable to formatted records only and will scroll the display to a nominated field within the default record only.

If the field is an array of repeating elements, then a field subscript must be supplied in parentheses "( )" identifying the individual array field element to be located. Array field co-ordinates must be specified for each dimension in the array where each co-ordinate separated is separated from the next by a comma (,). e.g. #12(1,2) identifies an element of a 2-dimensional array.

field_name
The name of a field in the default record. *field_name* may be fully qualified, partially qualified or unqualified (see Field Value expression terms), and may include a subscript array element reference. e.g. Member(3)

Regardless of whether the ABBREVIATION option has been set on, the group name or field item designators that constitute *field_name* may be abbreviated, starting with the first letter of the designator. If *field_name* identifies more than one field, then only the first occurrence of a field that matches *field_name* will be located.

field_ref
A field reference number in the default record. *field_ref* may include a subscript array element reference. e.g. #22(3)

FIRST
Search forwards from the first field in the default record to locate the first field that matches the specified field name or reference.

LAST
Search backwards from the last field in the default record to locate the last field that matches the specified field name or reference.

NEXT
Search forwards from the field that is currently displayed first in the visible display, to locate the next field that matches the specified field name or reference.

PREV
Search backwards from the field that is currently displayed first in the visible display, to locate the previous field that matches the specified field name or reference.

Examples:

```
locate 3
```
Scroll the display area so that the line that is 3 lines below the focus line becomes the new current line.

```
-18
```
Scroll the display area so that the line that is 18 lines above the focus line becomes the new current line.

```
:18
```
Scroll the display area so that the visible record at line number 18 becomes the new current line.

```
locate  (LastName = 'Jones' &  Salary > '21950') | (Dept \>> 'Tech')  prev
```
Scroll the display area so that the first record before the focus line that is of the default record type and also satisfies the specified expression, becomes the new current line.

```
locate  #15
```
Scroll to field number 15 in the default record.

See Also:

DOWN
FIND
UP
WHERE

MACRO

Syntax:

```
>>--- MACRO ---- macro_name ---+---------------+-----------------------------><
                               |               |
                               +- parm_string -+
```

Description:

Force SDE to execute the REXX macro specified by *macro_name* as opposed to an SDE command of the same name.

By default, SDE first checks a user supplied token for a recognised SDE command. If the token is not recognised as a command, an attempt is made to find a macro with the same name. Therefore, when invoking a macro, the command verb MACRO may not be necessary.

Any text specified following the macroname is passed to the macro as a parameter string. The specified macro is read into memory from disk, executed and removed from memory on completion.

If the macro cannot be located, an error message is issued and the MACRO command fails.

Parameters:

macro_name
: Name of the macro to be executed.

  If *macro_name* is a full fileid containing file name, path, etc., then the macro is loaded from the specified location. If only a file name is specified, each directory in the macro path is searched for a matching macro name.

parm_string
: Text to be passed to the macro as a parameter string.

Examples:

`macro cbl.dist.cbli.site.cble(xarc) XXFLD 22 33`
: Execute the macro XARC in library CBL.DIST.CBLI.SITE.CBLE with parameters "XXFLD 22 33".

`macro select`
: Execute the user macro SELECT from a library in the macro path, not the SDE command SELECT.

MAP

Syntax:

```
>>---+-- MAP --+-------------------------------------------------------------><
     |          |
     +-- FMT --+
```

Description:

Display records or record segments in single record, formatted view.

If the record-type (RTO) assigned to each record or record segment has been disabled (record data is unformatted), then MAP will re-enable the assigned record-types to reformat the record data.

Compare with FORMAT SINGLE which does not attempt to re-enable the assigned record-types.

Parameters:

MAP has no parameters.

See Also:

FORMAT
CHAR
UNFMT
VFMT
ZOOM

MARK

Description:

SDE CLI command, MARK, performs the same operation as the CBLe CLI command MARK. See MARK in CBLe Text Edit documentation.

MORE

Syntax:

```
>>--- MORE ---+-- where_clause --+-------------------------------------------><
               |                  |
               +-- line_flag -----+
```

Description:

Make visible any EXCLUDED records that are of the default record type and also satisfy the specified *where_clause* criteria. The current line of the display and any records that are already visible are unaffected by the MORE command.

If MORE is executed with no parameters, then all EXCLUDED records that are of the default record type are made visible.

The record type used on a MORE command becomes the new value of the DRECTYPE option.

Parameters:

where_clause
where_clause syntax is described fully in documentation for the WHERE command.

line_flag
line_flag may be specified to locate a record or record segment that has been flagged with the requested line flag.

Each valid *line_flag* keyword detailed below corresponds to a built-in function.

A description of each *line_flag* keyword may be found in its equivalent function description.

***line_flag* Keyword**	**Built-in Function**
ERRor	CHANGEERROR()
CHAnge | Chg	CHANGEOK()
ALTered | UPDated	CHANGED()
DATaerror	DATATYPEERROR()
DUPkey	DUPLICATEKEY()
EMPTY	EMPTYSLOT()
EXcluded | X	EXCLUDED()
LABel	HASPOINT()
COMmand | CMd	HASPREFIXCMD()
IDentify | IDrequired	IDREQUIRED()
NEW	INSERTED()
KEYChanged | KEYChg	KEYCHANGED()
LENgtherror	LENGTHERROR()
EOL | NOEOL	NOEOL()
SAVE	SAVEREQUIRED()
TRNC | TRUNCated	TRUNCATED()
VALERRor	VALUEERROR()

Examples:

```
more    #5 >> X'41'
```

Make visible any EXCLUDED records that are of the default record type where field reference number 5 begins with X'41' (ASCII character 'A').

`more LastName = C'Evans'`
Make visible any EXCLUDED records that are of the default record type where the contents of field "LastName" is equal "Evans".

See Also:

FLIP
LESS
WHERE

MSG

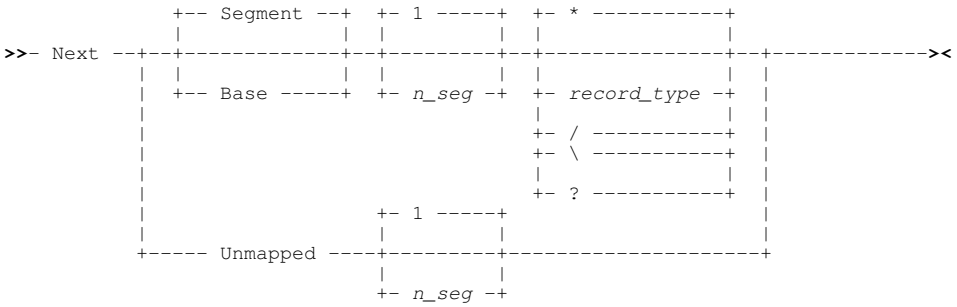
Description:

SDE CLI command, MSG, performs the same operation as the CBLe CLI command MSG. See MSG in CBLe Text Edit documentation.

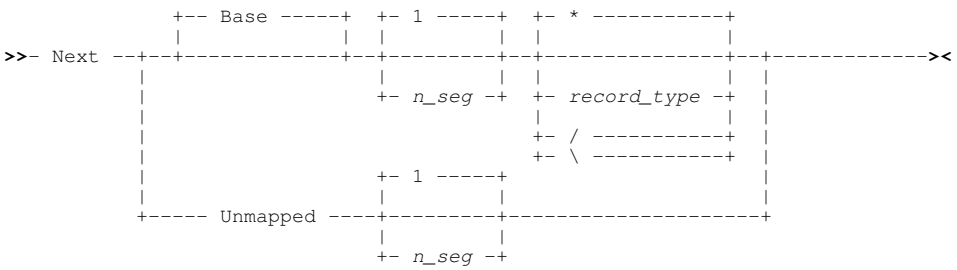
NEXT

Syntax:

Segmented Record Edit:



Non-Segmented Record Edit:



Description:

When used in an SDE BROWSE/EDIT view of segmented records, NEXT scrolls to a segment that follows the current segment, making it the new current segment. For non-segmented record BROWSE/EDIT, every record is considered to be a primary (base) segment with no secondary segments. Therefore, using NEXT SEGMENT to scroll to secondary segments is invalid and will always return ZZSD441E error message.

In multi-record view, the view of the data within the SDE window is scrolled down to the next selected segment. In single record view, NEXT will display the next selected segment.

NEXT scrolls only to segments of a specific type (primary, secondary or unmapped) and, optionally, segments assigned a specifically named record type (RTO). In contrast, DOWN (in multi-record view) and RIGHT (in single-record view) scroll to records or record segments that follow, regardless of segment type and assigned RTO.

NEXT SEGMENT (the default for segmented record display) restricts scrolling to only those secondary segments belonging to the current segmented record. If the specified secondary segment cannot be found within the current record (e.g. NEXT 6 for a record containing only 5 secondary segments), then the last segment to match the NEXT parameter criteria becomes the current segment and the following message is returned:

```
ZZSD441E Base record exhausted. Final match (#nn of mm) displayed.
```

Where *nn* is the last occurrence of the segment assigned record type *record_type* to be found following the current segment, and *mm* is the requested number of matching segments to be scrolled (*n_seg*).
If using the default parameter values to simply scroll to the next secondary segment, this message will contain `(#0 of 1)` where an attempt has been made to scroll to the first secondary segment following the last segment in the record.

Parameters:

`SEGMENT | BASE | UNMAPPED`
: Specifies the type of segment to be selected as the current segment.

  SEGMENT indicates a secondary segment, BASE indicates a primary (base) segment and UNMAPPED indicates an unformatted (primary or secondary) segment. Note that unformatted primary segments (i.e. records) have record type "Unmapped" and unformatted secondary segments have record type "UnmappedSeg".

  Default is SEGMENT for segmented record browse/edit and BASE otherwise.

n_seg
: Specifies the occurrence of a segment following the current segment that matches the optionally specified *record_type*. Default is 1.

`* | record_type | / | \ | ?`
: Specifies the record type (RTO) of the segment that is to become the new current segment. Only segments assigned this record type will be included in the count of segments to be scrolled.

  "*" (asterisk) indicates that segments may be of any record type. *record_type* nominates a specific record type. "/" or "\" indicates that segments are to be of the record type assigned to the current segment. "?" indicates that segments are to be of the record type **not** assigned to the current segment.

  Default is *.

Examples:

`NEXT`
: For segmented record browse/edit, scroll to the secondary segment within the same record that immediately follows the current segment. Fot non-segmented browse/edit, scroll to the next record.

`N B`
: Scroll to the next primary segment (record).

`N B 2 /`
: Scroll forwards to the 2nd occurrence of a primary segment (record) which is assigned the same record type as the current segment.

`N U`
: Scroll forwards to the next unformatted primary or secondary segment.

See Also:

PREVIOUS
LEFT
RIGHT

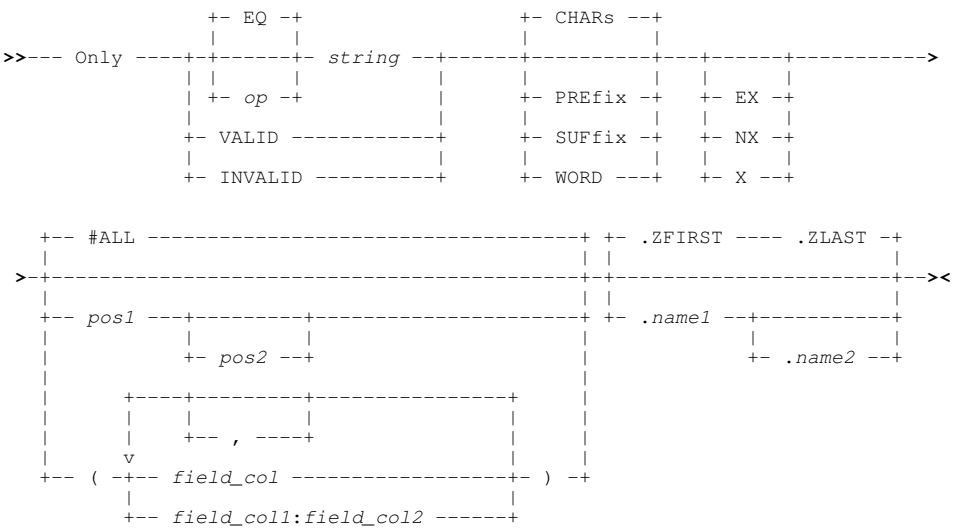
NOMSG

Description:

SDE CLI command, NOMSG, performs the same operation as the CBLe CLI command NOMSG. See NOMSG in CBLe Text Edit documentation.

ONLY

Syntax:



Description:

Based on the FIND ALL command, ONLY displays all records or record segments assigned the default record type that satisfy the specified search *string* criteria. All records or segments of the default record type that do **not** satisfy the search *string* criteria are excluded.

Note that the display of records or segments that are not assigned the default record type is unaffected.

All occurrences of the search string or numeric value are highlighted in the text. (Enter the RESET FIND command to turn off the highlighting.)

When the execution time of an ONLY command exceeds 1 second, a progress window is opened displaying the number of records processed so far. This window also provides the user with the oportunity to interrupt the operation using the Attn key.

The FORMAT of the SDE display affects the execution of ONLY.

FORMAT CHAR and HEX:

A character compare for the supplied search string is performed against entire, unformatted data records.

field_col and #ALL parameters are not applicable to these display formats and are ignored.

FORMAT SINGLE and TABLE:

For fromatted records, the search string is compared against **individual fields** that have been selected for display in the expanded data record. (See SELECT)

Fields are searched from left to right in the order that they appear in the display. This is true regardless of the order in which field columns are specified on the ONLY command, or the order in which fields are encountered within the expanded record.

If a field column is specified on the ONLY command that is not part of the display (e.g. an expanded group field or a field that has been removed from display by a SELECT command), then an error message is returned.

For character data type fields, a string compare is performed. For numeric data type fields (binary, packed decimal, floating point, zoned, etc.), then the following will occur:

1. If *pos1*, *pos2* positional parameters are **not** specified, the search string is interpreted as a signed numeric value and an arithmetic compare for equality is performed against the field's formatted numeric value.

   The length and data type of the numeric field, and the number of digits in the search value are not significant. e.g.

   ```
   ONLY  67
   ```

   Finds numeric fields with value "67" (e.g. "0067", "67.00", "0.0670E+03") and character fields containing the string "67" (e.g. "167 Baker Street").

   If the search string is non-numeric, then numeric data type fields are not searched. Therefore, to bypass searching numeric data type fields, explicitly set the search string to be character or hex using 'string', C'string' or X'string' formats

respectively. e.g.

```
ONLY   '67'
ONLY  C'67'
ONLY  X'F6F7'
```

Finds only character fields containing the string "67".

2. If *pos1*, *pos2* positional parameters are specified, the search string is interpreted as being a character string and a string compare is performed on the character representation of numeric data.

   Although the range of positions may span a number of fields, the search is still performed against individual fields within the range. i.e. a match for the search string will not be found if the data spans a field boundary.

   A match for the search string on just part of the character representation of a numeric field is considered to be a successful hit and the whole numeric field is highlighted. e.g.

   ```
   ONLY   476    21  100
   ```

   Finds character and numeric fields where character representation of the data contains the string "476". (e.g. a zoned decimal field with value "14760" or "-4762")

Where a character string or numeric value is found in a numeric field, the entire field is highlighted and the cursor positioned at the start of the field.

Parameters:

op

A relational operator used in the compare operation which determines the relationship that the data must have with the ONLY search string in order for it to be identified as a successful match.

Valid values for *op* are as follow:

Operator	Description
EQ	Data must be equal to *string*. (Default)
NE	Data must be not equal to *string*.
GT	Data must be greater than *string*.
GE	Data must be greater than or equal to *string*.
LT	Data must be less than *string*.
LE	Data must be less than or equal to *string*.

If a character string compare is performed, the EBCDIC values assigned to characters in the search and data strings determine the relationship (equal to, greater than or less than) between the two strings.

string

The ONLY search string. The search string may be any of the following:

◊ An unquoted numeric value. The search string is treated as a numeric value when a numeric field is searched in a SINGLE or TABLE format view. In all other cases a numeric search string is treated as a character string.

◊ An unquoted character string containing no commas or blanks. The search for the character string will be case-insensitive so that uppercase and lowercase characters are treated as being the same.

◊ A character string enclosed in single (') or double (") quotation marks. The search string may contain embedded commas and blanks and the character string will be case-insensitive.

Two adjacent quotation mark characters that are embedded in a search string which is enclosed by the same quotation mark characters, will be treated as a single occurrence of the character. e.g.

```
ONLY  'Jim O''Brien'
```

Find the character string "Jim O'Brien".

◊ A character string enclosed in single (') or double (") quotation marks with the prefix C. This is equivalent to specifying a quoted search string but that the string search will be case-sensitive. (e.g. C'Book')

◊ A hexadecimal string enclosed in single (') or double (") quotation marks with the prefix X.

◊ A picture string enclosed in single (') or double (") quotation marks with the prefix P.

Picture strings use special characters to represent a generic group of characters as described below. Any character in a picture string that is not one of these special characters is untranslated.

String	Description
P'='	Any character.
P'¬'	Any non-blank character.
P'.'	Any non-displayable character.

P'#'	Any numeric character, 0-9.
P'-'	Any non-numeric character.
P'@'	Any uppercase or lowercase alpha character.
P'<'	Any lowercase alpha character.
P'>'	Any uppercase alpha character.
P'$'	Any non-alphanumeric special character.

VALID
Intended for use with formatted records, VALID will search fields for valid data. i.e. data that satisfies the field's assigned data type.

INVALID
Intended for use with formatted records, INVALID will search fields for invalid data. i.e. data that does not satisfy the field's assigned data type.

CHARS
For non-numeric search strings only, CHARS indicates that a successful match occurs if the search string is found anywhere within the data being searched.

PREFIX
For non-numeric search strings only, PREFIX indicates that a successful match only occurs if the search string is found at the start of a word within the data being searched. i.e. the matched text must precede an alphanumeric character and either be preceded by a non-alphanumeric character or be at the start of a line or field.

SUFFIX
For non-numeric search strings only, SUFFIX indicates that a successful match only occurs if the search string is found at the end of a word within the data being searched. i.e. the matched text must be preceded by an alphanumeric character and either precede a non-alphanumeric character or be at the end of a line or field.

WORD
For non-numeric search strings only, WORD indicates that a successful match only occurs if the search string is found to be a complete word within the data being searched. i.e. the matched text must either be preceded by a non-alphanumeric character or be at the start of a line or field, and either precede a non-alphanumeric character or be at the end of a line or field.

EX
X
Search EXCLUDED data records only.
ONLY does not search records that are NOTSELECTED or SUPPRESSED.

NX
Search only visible data records (i.e. not EXCLUDED).
ONLY does not search records that are NOTSELECTED or SUPPRESSED.

pos1
The first position of a range of positions within the data record to be searched.

For SINGLE and TABLE formats, this is a position in the expanded record. Only those fields, or parts of fields, that fall within the position range will be searched. Fields will be searched in the order that they occur within the display area.

pos1 may be a positive or negative integer value (not zero) and must be a value that is less than or equal to the maximum length of the data records or, for formatted record data, the length of the expanded record.

A negative value represents a position in the record relative to the end of the record. Therefore, where position 1 references the 1st character in the record, position -1 references the last character.

For all display formats, the string is treated as being non-numeric and the search is actioned on the character representation of the record data.

pos2
The last position of a range of positions within the data record to be searched.

Like *pos1*, *pos2* may be a positive or negative integer value (not zero). If *pos1* references a position within the record data which is higher than that referenced by *pos2*, then the *pos1* and *pos2* values are swapped.

If *pos2* is greater than the maximum length of the data records or, for formatted record data, greater than the length of the expanded record, then *pos2* is set equal to the maximum (or expanded) record length.

Default is *pos1* plus the length of the search string minus 1.

#ALL
Search all field columns in the current TABLE or SINGLE format display.

field_col
An individual field column to be searched within a TABLE or SINGLE format display.

The field column may be identified by its reference number (e.g. #6) or its name (e.g. JobID). If a field name is used, enclosing parentheses are **mandatory** Field names are automatically converted to a field reference and Referencing the same field column more than once will not cause an error.

If the field is an array, then all elements of the array are searched. To search an individual element of an array, the element occurrence should be specified in parentheses as a subscript to the field reference/name. e.g. #13(3) for the 3rd element of a single dimension array. An entry must exist for each dimension of the array. e.g. RoomSize(6,2,4) - a three dimensional array.

Specification of multiple field columns must either be enclosed in parentheses and/or separated by commas. The field columns may be specified in any order, however, the data is always searched from the first column in the display to the last.

Field column search specifications may be a combination of individual field columns and columns ranges. e.g. (JobID #6 Tax_Reference #12 #15:#20).

A search is only performed on field columns that are selected for display. Therefore any field column specified on the ONLY command that has not been selected for display, will be ignored.

field_col1:*field_col2*
The first and last fields of a range of field columns to be searched within a TABLE or SINGLE format display.

field_col1 and *field_col2* must be separated by ":" (colon) and if *field_col1* occurs after *field_col2* in the data record, then the values are swapped. *field_col1* and *field_col2* have the same specifications as *field_col*.

Specification of multiple field column ranges must either be enclosed in parentheses and/or separated by commas. Field column search specifications may be a combination of individual field columns and field columns ranges. e.g. (FirstName:LastName, #2, Salary:Bonus, EmpNo, #10:#15). If field names are used, enclosing parentheses are **mandatory**.

.name1
A label name identifying the first record of a range of data records to be searched. The preceding "." (dot) is mandatory. Default is .ZFIRST.

.name2
A label name identifying the last record of a range of data records to be searched. The preceding "." (dot) is mandatory. If *.name2* occurs before *.name1* in the display, then the order is reversed.

See Also:

EXCLUDE
FIND
SELECT

OPTIONS

Description:

OPTIONS displays the current setting of all SDE options in a message window.

OPTIONS is equivalent to executing QUERY *option* for every SET/QUERY/EXTRACT option supported by SDE.

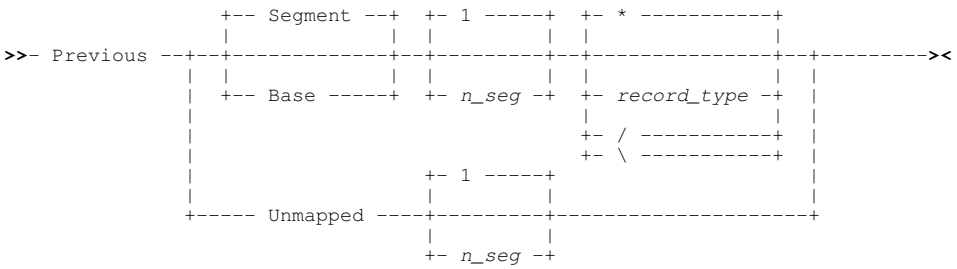
Parameters:

OPTIONS has no parameters.

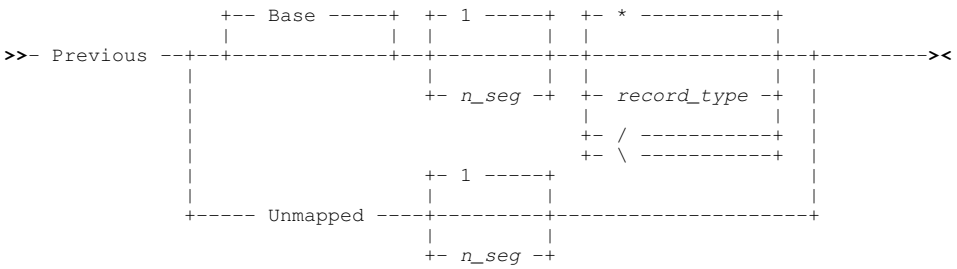
PREVIOUS

Syntax:

Segmented Record Edit:



Non-Segmented Record Edit:



Description:

When used in an SDE BROWSE/EDIT view of segmented records, PREVIOUS (or PREV) scrolls to a segment that occurs before the current segment, making it the new current segment. For non-segmented record BROWSE/EDIT, every record is considered to be a primary (base) segment with no secondary segments. Therefore, using PREV SEGMENT to scroll to secondary segments is invalid and will always return ZZSD441E error message.

In multi record view, the view of the data within the SDE window is scrolled upwards to a previous segment. In single record view, PREV will display a previous segment.

PREV scrolls only to segments of a specific type (primary, secondary or unmapped) and, optionally, segments assigned a specifically named record type (RTO). In contrast, UP (in multi-record view) and LEFT (in single-record view) scroll to preceding records or record segments, regardless of segment type and assigned RTO.

PREV SEGMENT (the default for segmented record display) restricts scrolling to only those secondary segments belonging to the current segmented record. If the specified secondary segment cannot be found within the current record (e.g. PREV 6 for a record containing only 5 secondary segments), then the last segment to match the PREV parameter criteria becomes the current segment and the following message is returned:

```
ZZSD441E Base record exhausted. Final match (#nn of mm) displayed.
```

Where *nn* is the last occurrence of the segment assigned record type *record_type* to be found previous to the current segment, and *mm* is the requested number of matching segments to be scrolled (*n_seg*).
If using the default parameter values to simply scroll to the previous secondary segment, this message will contain `(#0 of 1)` where an attempt has been made to scroll to the first secondary segment prior to the first secondary segment in the record.

Parameters:

`SEGMENT | BASE | UNMAPPED`
: Specifies the type of segment to be selected as the current segment.

  SEGMENT indicates a secondary segment, BASE indicates a primary (base) segment and UNMAPPED indicates an unformatted (primary or secondary) segment. Note that unformatted primary segments (i.e. records) have record type "Unmapped" and unformatted secondary segments have record type "UnmappedSeg".

  Default is SEGMENT for segmented record browse/edit and BASE otherwise.

n_seg
: Specifies the occurrence of a segment preceding the current segment that matches the optionally specified *record_type*. Default is 1.

`* | record_type | / | \`

Specifies the record type (RTO) of the segment that is to become the new current segment. Only segments assigned this record type will be included in the count of segments to be scrolled.

"*" (asterisk) indicates that segments may be of any record type, *record_type* nominates a specific record type, and "/" or "\" indicates that segments are to be of the record type assigned to the current segment.

Default is *.

Examples:

`PREV`
For segmented record browse/edit, scroll to the segment within the same record that immediately precedes the current secondary segment. Fot non-segmented browse/edit, scroll to the previous record.

`P B`
Scroll to the first primary segment (record) that precedes the current segment.

`P B 2 /`
Scroll backwards to the 2nd occurrence of a primary segment (record) which is assigned the same record type as the current (primary) segment.

`P U`
Scroll backwards to the 1st unformatted primary or secondary segment that precedes the current segment.

See Also:

NEXT
LEFT
RIGHT

QQUIT

Syntax:

```
>>--- QQuit ------------------------------------------------------------------><
```

Description:

Close the current SDE window view only.

If additional SDE views exist for the data, then these will remain open.

If only one SDE edit view exists for the data, then any unsaved changes to the data will be discarded. However, if a non-temporary structure (SDO) is used to map the record data and changes have been made to that structure during the course of the edit session (e.g. USE WHEN), then the user will be prompted to save the changed structure to its structured data file (SDF). See command, SAVESTRUCTURE.

Parameters:

QQUIT has no parameters.

See Also:

END
CANCEL

QUERY

Description:

See SET/QUERY/EXTRACT Options.

RCHANGE

Syntax:

Description:

Repeat the find and replace performed by the last CHANGE command. The RCHANGE search will be executed on records that are of the same record type as that used by the last CHANGE command.

Where the last CHANGE issued was CHANGE LAST or CHANGE PREV, RCHANGE will perform a CHANGE PREV operation, otherwise RCHANGE performs a CHANGE NEXT operation. i.e. Search forwards (NEXT) or backwards (PREV) from the current cursor location to find the next or previous occurrence of the string/value respectively.

If the cursor is not within the window's data display area, the forwards or backwards search begins at the first position of the first visible or excluded record within the display area that is of the record type used by the last CHANGE.

RCHANGE is assigned to function key **PF6** by default.

Parameters:

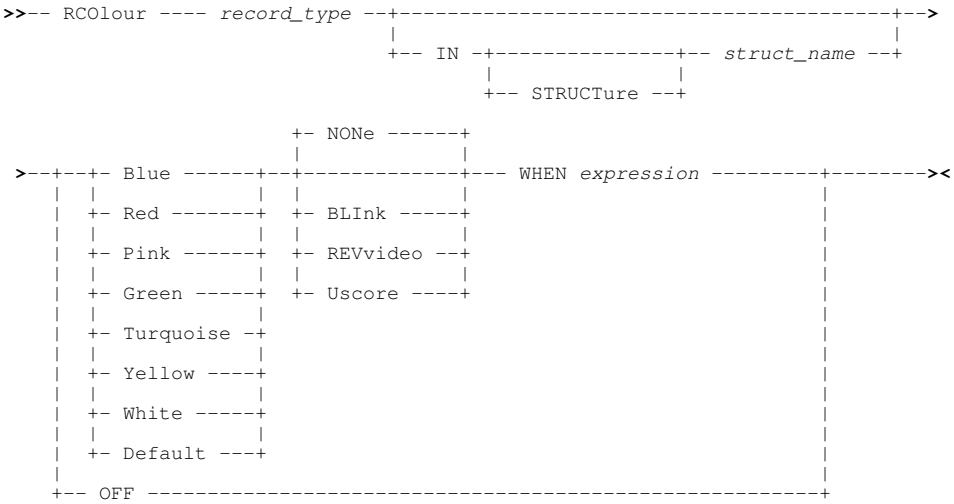
RCHANGE has no parameters.

See Also:

CHANGE
EXCLUDE
FIND
RFIND
Find and Replace Data

RCOLOUR

Syntax:



Description:

RCOLOUR may be used to apply preferred colouring to records assigned specific record types (RTO) based on selection criteria.

Record colouring definitions specified by the RCOLOUR command are stored in the record type definition within the relevant structure (SDO). If not already loaded, the required SDO is loaded from its structured data file (SDF) and updated accordingly. If a non-temporary SDO, then this change to the SDO may be saved (written back to the SDF) by the user when the structure is dropped or on execution of the SAVESTRUCTURE command.

Any number of RCOLOUR specifications may exist for an individual record type based on different WHEN *expression* criteria. If records assigned this record type satisfy the WHEN *expression* criteria of more than one RCOLOUR definition, then the RCOLOUR definition used is the one that was defined first.

Parameters:

record_type
The name of the record type (RTO) for which the colour definition will be applied.

IN (STRUCTURE) *struct_name*
The name of the structure (SDO) in which *record_type* is defined.
Default is to the current (i.e. last referenced) structure. This is the structure used in the current SDE window view or explicitly referenced by an SDE command (e.g. USE WHEN)

BLUE | RED | PINK | GREEN | TURQUOISE | YELLOW | WHITE | DEFAULT
Supported colours. If DEFAULT is specified, the default colour for record display is used.

BLINK | REVERSE | USCORE | NONE
Extended highlighting. The colour may blink, be displayed in reverse video or be underlined. Default is NONE.

WHEN *expression*
Specifies the criteria that must be satisfied before the defined colouring will occur.

expression is a valid SDE expression which supports function calls, *record_type* field names and references, sub-expressions, arithmetic, relational and logical operators. The result of the WHEN expression must be numeric and is treated as being Boolean in nature with a zero value indicating a "false" condition and any non-zero value indicating a "true" condition.

The WHEN expression is applied to each record assigned the record type *record_type* and, if the result is "true", the specified record colouring is applied. To apply record colouring for all records of record type *record_type*, simply specify "WHEN 1".

OFF
Remove all specified record type (RTO) colouring from the SDO definition, reverting back to the default record colouring settings.

Examples:

<sd rcol SRC in CBL.SDO(ADATA) red uscore when (substr(#21,2) >> 'USING')
Define colouring of record data assigned to record type "SRC" in structure "CBL.SDO(ADATA)" when data starting in position 2 of field number 21 begins with "USING".

See Also:

SET/QUERY/EXTRACT COLOUR, COLOR
DROP
SAVESTRUCTURE
USE

RECLEN

Syntax:

```
>>-- RECLength ---+------------+-----------------------------------------><
                  |            |
                  +-- ON ------+
                  |            |
                  +-- OFF -----+
```

Description:

RECLENGTH controls the display of the Record Information Length column.

If the records are variable length and Full Edit, KSDS Edit or Auxiliary Edit is in effect, then the length of each record displayed in this column may be altered simply by overtyping the existing value.

If the length of a record is increased, then pad characters are appended to the record. The pad character used is that defined by the current value for PAD. Reducing a record's length will truncate record data.

If a structure has been applied to the edited records, then the record type (RTO) associated with the updated record is re-evaluated and the contents of the SDE edit view updated accordingly.

Parameters:

ON
OFF

Set display of the record Length column on or off.
RECLENGTH ON and RECLENGTH OFF is equivalent to SET RECINFO ON LENGTH and SET RECINFO OFF LENGTH respectively.

Default is to toggle between RECLENGTH ON and RECLENGTH OFF.

See Also:

LENGTH - SET/QUERY/EXTRACT
RECINFO - SET/QUERY/EXTRACT
PAD - SET/QUERY/EXTRACT

REDO

Syntax:

```
>>-- REDO ------------------------------------------------------------------><
```

Description:

REDO re-applies one level of change made to the current file that was previous undone by the last execution of the UNDO command. Where UNDO has not previously been executed, REDO has no affect.

Each change level corresponds to an update of a data field within a record.

REDO is assigned to PF23 by default and may be executed repeatedly to re-apply multiple levels of changes.

Following one or more executions of UNDO, REDO will recover the changes even after the following have occured:

- The cursor position changes.
- The file is saved.
- Changes are made to other files in the file edit ring.

However, REDO is not able to recover changes following an UNDO if further changes are made to record data. (e.g. fields updated, records deleted or inserted.)

The third number following "Alt=n,n;m" on the status line indicates the number of change levels. If this number is followed by an "*" (asterisk), then it is possible to REDO previous UNDO commands.

Parameters:

None.

See Also:

UNDO
UNDOING - SET/QUERY/EXTRACT
Undo and Redo Changes

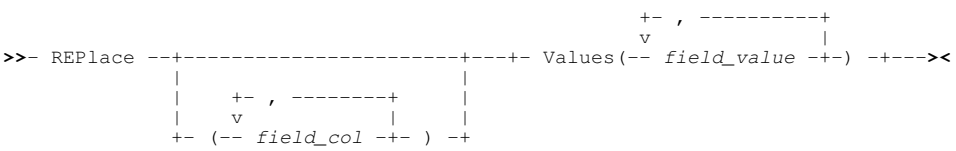
REFRESH

Description:

SDE CLI command, REFRESH, performs the same operation as the CBLe CLI command REFRESH. See REFRESH in CBLe Text Edit documentation.

REPLACE

Syntax:



Description:

Replace data in fields within the record occupying the focus line.

The record occupying the focus line must not be a shadow line and must contain any specified fields references.

REPLACE is primarily used within SDE Edit REXX macros.

Parameters:

`( field_col,... )`
A list of field columns into which corresponding *field_value* values are to be inserted.

The field column may be identified by its reference number (e.g. #1) or its name (e.g. SeqNUM). Specification of multiple field columns must be separated by commas and enclosed in parentheses.

If the *field_col* reference is a **struct** or a **union**, then an error message is returned. If the *field_col* reference is an array, then an individual array element must be specified in parentheses as a subscript to the field. e.g. #13(3) for the 3rd element of a single dimension array. A subscript entry must exist for each dimension of the array. e.g. RoomSize(6,2,4) - an element within a three dimensional array.

If a list of field columns is not specified then as many of the set of selected fields in the current view as there are values in the list of field values, are used as the default. In this case, the order of the field columns is equal to the order in which fields are displayed in the current view.

`VALUES ( field_value,... )`
A list of field values to be inserted into corresponding *field_col* entries in the field column list.

If one or more *field_col* entries are specified (i.e. a field list is specified), then there must be the same number of *field_value* values to correspond with each *field_col* entry.

If a field value contains special characters, blanks or commas, then it should be enclosed in quotes.

Examples:

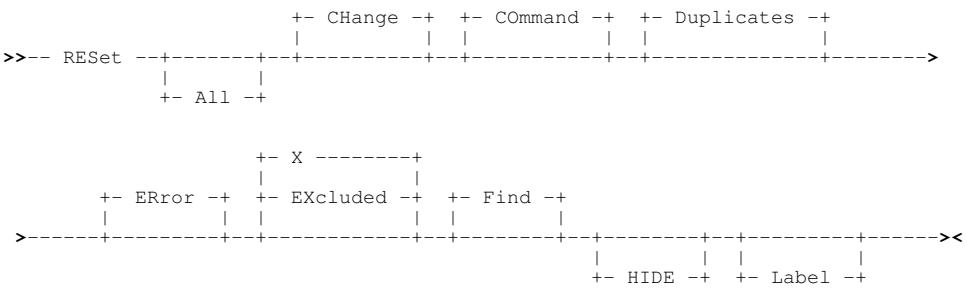
```
replace    (NAME,#5,BONUS)    values ("Tom Jones", 62, 200.00)
```
Replace existing values in the specified fields with those supplied. Numerical data will be converted to the appropriate data type for the field.

See Also:

INSERT

RESET

Syntax:



Description:

Restores aspects of the current SDE BROWSE or EDIT window display to their original settings. Only data records that are of the default record type are affected by the RESET command.

Parameters:

ALL
: Equivalent to: `RESET COMMAND DUPLICATES EXCLUDED FIND LABEL CHANGE ERROR`

CHANGE
: Remove "==CHG>" flags from the prefix area.

COMMAND
: Remove any pending prefix area commands.

DUPLICATES
: Reset the "=DUPE>" and "=DUPF>" line flags introduced with support for KSDS data set editing. These flags are set when an attempt is made to save changes to the data and cause the save operation to fail.
"=DUPE>" indicates that a duplicate key exists between 2 or more, new or changed records within the edit session.
"=DUPF>" indicates that a duplicate key exists between a new or changed record in the edit session and an existing file record.

ERROR
: Remove "==ERR>" and "=ERRV>" line flags from the prefix area.
"=ERRV>" line flag indicates that the expanded record has a variable length or variable array size value error.

EXCLUDED
X
: Restore (make visible) any data records belonging to the default record type that have been previously excluded from view.

FIND
: Remove all highlighting of search strings and numerical fields that have been found by the FIND command.

HIDE
: Redisplay all shadow lines that have beed removed from the display by the HIDE command.
RESET HIDE is equivalent to SET SHADOW ON ALL.

LABEL
: Remove all line label names set via the SET POINT command or entered in the prefix area directly.

RFIND

Syntax:

```
>>-- RFInd ------------------------------------------------------------------><
```

Description:

Repeat the search performed by the last FIND command. If RFIND is executed following an EXCLUDE or CHANGE command, then the search string used will be that specified on the EXCLUDE or CHANGE command. The RFIND search will be executed on records that are of the same record type as that used by the last FIND, EXCLUDE or CHANGE command.

Where the last FIND issued was FIND LAST or FIND PREV, RFIND will perform a FIND PREV operation, otherwise RFIND performs a FIND NEXT operation. i.e. Search forwards (NEXT) or backwards (PREV) from the current cursor location to find the next or previous occurrence of the string/value respectively.

If the cursor is not within the window's data display area, the forwards or backwards search begins at the first position of the first visible or excluded record within the display area that is of the record type used by the last FIND.

RFIND is assigned to function key **PF5** by default.

Parameters:

RFIND has no parameters.

See Also:

CHANGE
EXCLUDE
FIND

RIGHT

Syntax:

```
>>- Right---+--------------------+-------------------------------------------><
            |                    |
            +-- Cursor ----------+
            |                    |
            +-- CSR -------------+
            |                    |
            +-- Data ------------+
            |                    |
            +-- Half ------------+
            |                    |
            +-- Max -------------+
            |                    |
            +-- Page ------------+
            |                    |
            +-- n_cols ----------+
```

Description:

In multi record view, scroll towards the right the display of field and header lines belonging to records that are of the default record type. The display of all other visible records, header lines and shadow lines remains unchanged.

In single record view, RIGHT will display the fields belonging to a visible data record that has a higher line number than the record (or record segment) that is in the current view. For display of segmented records, LEFT and RIGHT scroll through each segment in the file, regardless of the record to which it belongs. NEXT and PREV may be used to scroll between secondary segments of the current record only. RIGHT CURSOR/DATA/HALF/PAGE are not applicable in single record view.

RIGHT is assigned to **PF11** by default. Any characters specified on the command line when the PFKey is hit will be concatenated to the command and treated as a parameter string.
Where no parameter is specified, the scroll amount will be the value specified in the "**Scroll>**" field.

Multi Record View Scrolling:

Columns may have the HOLD flag set on by the SELECT command. These columns are static in the display and, like the prefix area and Lrecl column, are unaffected by LEFT and RIGHT scrolling. Columns that have the HOLD flag set off are referred to as **floating** columns.

The following interpretation of cursor location applies so that RIGHT CURSOR can operate successfully on the appropriate data:

- Cursor is located at an offset within a column header line.
  The cursor position is interpreted as being at the same offset within the data record that immediately follows the group of header lines.
- Cursor is located within a column of type character (AN) but beyond the width of the character data.
  The cursor position is interpreted as being the last position of the displayed character data.
- Cursor is located immediately to the left of a column of any type.
  The cursor position is interpreted as being the the first position of the data displayed within that column.

Where the first column to be displayed is not of type character (AN), the magnitude by which the display is scrolled to the right is always reduced to display all field data belonging to the first column. Similaraly, if the last column to be displayed is not of type character and the display width is too small to accomodate all the column's data and header text, the column will not be displayed.

No adjustment is made when scrolling right into an offset within a character data column. i.e. where the first column of the display is of type character, the first position of the column display need not be the first character of the character data. If the last column to be displayed is of type character and the display width is too small to accomodate all the column's data and header text, the column will be truncated.

Attempting to scroll right beyond the last data column will make the last data column the first column of the display. Furthermore, if the last data column is of type character (AN), then scrolling beyond this column will make the last character of the data the first character of the display.

Parameters:

`CURSOR`
`CSR`

The focus column becomes the first column of the scrolled display.
In addition to this, if the focus column is type character (AN) and the cursor is positioned at an offset into the focus column, then the character data at the cursor offset will occupy the first position within the first column of the scrolled display.

If any of the following conditions are true, then RIGHT PAGE is executed instead:

◊ Cursor is positioned anywhere other than within a floating column in a visible data record.
◊ Focus column is **not** type AN and is already the first floating column of the display.
◊ Focus column is type AN, is already the first floating column of the display and cursor position is at the first position of the displayed column data.

`DATA`

Scroll right so that the last floating column of the current display area becomes the first floating column of the scrolled display.
If this column is type character (AN) and the last position of the display falls on a character that is at an offset into the column data, then the character data at that offset will occupy the first position within the first column of the scrolled display.

`HALF`

Scroll a number of columns so that the column situated half way along the width of the current display of floating columns, becomes the first floating column of the scrolled display.
If this column is type character (AN) and the half display position falls on a character that is at an offset into the column data, then the character data at that offset will occupy the first position within the first floating column of the scrolled display.

`MAX`

Scroll right to display the last column of data in a multi record view. Where the display area is able to contain all data columns in the data record, the first floating column becomes the first floating column of the scrolled display. Otherwise, the last column of data becomes the last column of the scrolled display.

In single record view, the last visible data record is displayed.

`PAGE`

Scroll right so that the column of data to the right of the last floating column of the current display area, becomes the first floating column of the scrolled display.
If this column is type character (AN) and the last position of the display plus one falls on a character that is at an offset into the column data, then the character data at that offset will occupy the first position within the first floating column of the scrolled display.

`n_cols`

In a multi record view, scroll right a specified number of floating columns.
The floating column of data that is *n_cols* to the right of the first floating column becomes the new first floating column of the scrolled display.

In single record view, the visible data record that is *n_cols* records after the record currently in view, gets displayed.

Examples:

```
right half
```

Scroll records of the default record type in a multi record view display area, right by half a display area width minus the width of any columns flagged with HOLD.

See Also:

DOWN
LEFT
UP

SAVE

Syntax:

```
>>-- Save ------------------------------------------------------------------><
```

Description:

Save changes to the current data.

Parameters:

SAVE has no parameters.

See Also:

FILE
SAVEAS

SAVEAS

Syntax:

```
>>-- SAVEAS ----------------------------------------------------------------><
```

Description:

Applicable to structured data edit only (not DB2 table edit), SAVEAS prompts the user for a new fileid to be assigned to the data in the focus SDE edit view, and then saves the data to this file.

The fileid entered by the user may be an MVS DSN, PDS/PDSE library and member or HFS file path of a new file. If the file already exists, error ZZSD229E is returned. A PDS/PDSE library and member may only be specified if the currently assigned fileid is a PDS/PDSE library member.

If necessary SAVEAS will invoke the appropriate Allocate NonVSAM or Define VSAM data set dialog window to create the new data set with the same data set organisation as the currently assigned fileid.

SAVEAS is not valid for SDE edit techniques KSDS Edit and Update-in-place Edit where not all records are loaded into storage.

Parameters:

SAVEAS has no parameters.

See Also:

SAVE

SAVESTRUCTURE

Syntax:

```
>>-- SAVEStructure ---+-----------------+-------------------------------------><
                      |                 |
                      +-- struct_name --+
```

Description:

Save an in storage Structure Definition Object (SDO) to a Structure Definition File (SDF) on disk.

If changes have been made to the structure (e.g. USE WHEN or RCOLOUR criteria have been applied) and not saved, then the user will be automatically prompted to save the SDO following execution of the DROP command or when the structure is automayically dropped on exit from the parent CBLe frame window.

Parameters:

struct_name
The name of the non-temporary, in-storage structure to be saved.
Default is to save the current (i.e. last referenced) structure. This is the structure used in the current SDE window view or explicitly referenced by an SDE command.

SDATA

Description:

SDE CLI command, SDATA, performs the same operation as the CBLe CLI command SDATA. See SDATA in CBLe Text Edit documentation.

SDATA command is supported so that it may be tolerated in edit macros when the focus CBLe child window is an SDE edit/browse view.

SEGTYPE

Syntax:

```
>>-- SEGtype -----+-- Secondary -----------------+----------------------------><
                  |                              |
                  +-- Primary -------------------+
                  +-- Base ----------------------+
```

Description:

Applicable to full or auxiliary edit of segmented records only, SEGTYPE is used to change the type (primary or secondary) of the focus segment. Note that Update-in-place edit does not support SEGTYPE.

SEGTYPE PRIMARY or SEGTYPE BASE changes a secondary segment to be a primary (base) segment so splitting the record into two. The record generated by the new primary segment is remapped so that all its segments are assigned an appropriate record type definition.

SEGTYPE SECONDARY changes a primary segment to be a secondary segment so joining the record with the record before. The record is remapped so that all its segments are assigned an appropriate record type definition.

Note that no action is taken if SEGTYPE does not change the segment type of the focus segment.

SEGTYPE may be undone and subsequently redone using the UNDO (<PF22>) and REDO (<PF23>) commands.

Parameters:

SECONDARY
Sets the focus segment to be a secondary segment.

PRIMARY | BASE

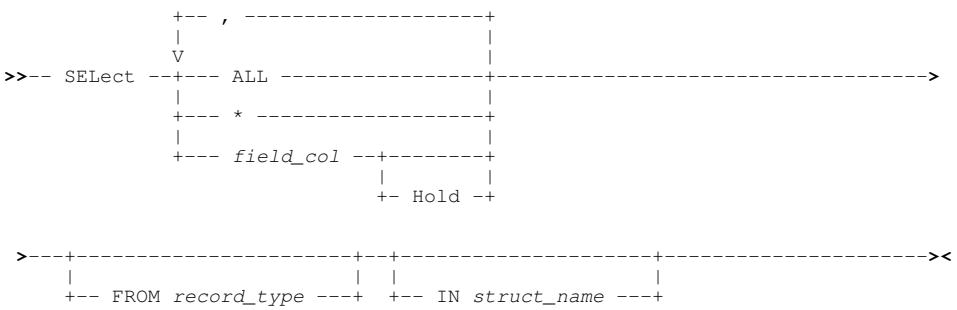
Sets the focus segment to be a primary (base) segment.

See Also:

SET/QUERY/EXTRACT FOCUS

SELECT

Syntax:



Description:

For a particular record type, specify the names of the fields to be displayed and also the order in which they are to be displayed within the current window view.

The selected field names must belong to the specified record type. If *record_type* is not specified, then the default record type is assumed.

If *struct_name* is not specified, then the SELECT command operates on the data in the SDE current window.

SELECT operates at the SDE window view level and so will not apply to additional views of the same SDE edited record data.

Parameters:

field_col
: Any field reference number or name defined for this record type. If *field_col* refers to a **struct**, then all elements of that structure are considered to be selected.

Hold
: Indicates that this field (and any others preceding it) is to be held visible when scrolling occurs.

ALL
: Shorthand to define all fields in their default order. Must be supplied as the last parameter.

* (asterisk)
: Shorthand to define all fields other than those explicitly defined already (i.e. the rest) in their default order. Must be supplied as the last parameter.

FROM *record_type*
: Defines the record type to which the selection applies. If *record_type* is not specified, then the default record type is used.

IN *struct_name*
: Defines the name of the structure definition object (SDO) to which the selection applies. The SDE window containing data mapped by this SDO will be the target window for this SELECT command. Default is the SDE current window.

Examples:

```
select from Employees  EmpNumber,Dept,LastName,FirstName
select EmpNumber,Dept,LastName,FirstName H,*
```

SET

Description:

See SET/QUERY/EXTRACT Options.

SORT

Syntax:

```
>>-- SORT ---- KEY ------------------------------------------------------------><
```

Description:

Sort VSAM KSDS records in the SDE edit view.

Parameters:

`KEY`
: Supported for VSAM KSDS data sets only, sorts records into ascending key sequence. During an edit session of a VSAM KSDS data set, records may be moved, inserted, deleted or the the key field updated. This parameter allows the user to re-order the records in key sequence.

  All records in the display (visible, EXCLUDED, NOT SELECTED and SUPPRESSED) are included in a SORT KEY operation.

SYSCOMMAND, TSO, CMS

Description:

SDE CLI command, SYSCOMMAND, performs the same operation as the CBLe CLI command SYSCOMMAND. See SYSCOMMAND in CBLe Text Edit documentation.

TEDIT

Syntax:

```
>>-- TEdit -- cble_command ---------------------------------------------------><
```

Description:

Direct a command to the CBLe text edit Environment.

The TEDIT command allows CBLe commands to be issued from an SDE window command line or SDE edit macro. Compare this with the SELCOPY/i CLI command SDATA which allows SDE commands to be executed from any type of window so long as a CBLe frame window exists.

Parameters:

`cble_command`
: Any CBLe CLI command.
  The command is passed to the current CBLe window (the CBLe window view on which focus was last placed.)

Examples:

```
tedit   edit   DEV.OEM.CBL.JCL(CBLINS01)
```
Open a data set for CBLe text edit.

```
tedit   insert <sdata use IMP_REC when #12 >= 100
```
Insert a line of text into the current CBLe edit window. The line of text is a CBLe edit command that may be executed via the PF4 key (CMDTEXT).

TOP

Syntax:

```
>>-- Top ------------------------------------------------------------------><
```

Description:

Display the first page of data.
Equivalent to the UP MAX command.

Parameters:

None.

See Also:

BOTTOM
UP

UNDO

Syntax:

```
>>-- UNDO ----------------------------------------------------------------><
```

Description:

Undo one level of changes made to the current file.

Each change level corresponds to an update of a data field within a record.

UNDO is assigned to PF22 by default and may be executed repeatedly to undo multiple levels of changes.

The third number following "Alt=n,n;m" on the status line indicates the number of change levels. If this number is not zero, then changes may be undone using UNDO.

Parameters:

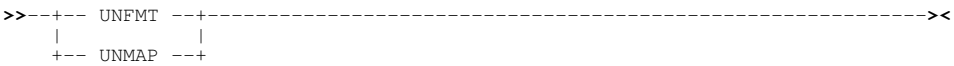
None.

See Also:

REDO
UNDOING - SET/QUERY/EXTRACT
Undo and Redo Changes

UNFMT

Syntax:



Description:

Display records or record segments in single record, unformatted character view by temporarily disabling the record-type (RTO) assigned to each record or record segment.

Compare with ZOOM of an SDE view set as FORMAT CHAR which displays the records/segments in a single-record, unformatted view but does not disable the assigned record-type.

Note that unformatted records have record type "UnMapped", unformatted segments of a segmented record have record type "UnMappedSeg".

Parameters:

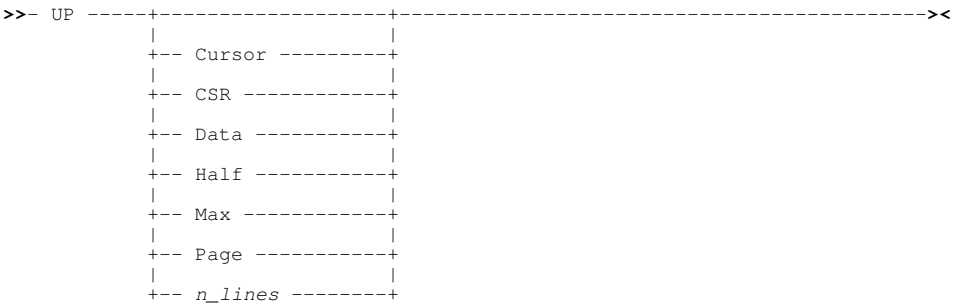
UNFMT has no parameters.

See Also:

FORMAT
CHAR
MAP (FMT)
VFMT
ZOOM

UP

Syntax:



Description:

Scroll the view of the data within the SDE window upwards towards the top of the file data.

Where no parameter is specified, the scroll amount will be the value specified in the **Scroll>** field in the top right corner of the window display.

Note that the first data record in any multi record view will **always** be preceded by its complete group of column header lines.

For single record views, the standard headers are always visible at the top of the display area and are not included as part of the scrollable text. Each non-header line within a single record view is identified as being a **field data line**.

UP and DOWN scroll commands will cause the window display area to be adjusted by a number of lines determined by the number of multi record view data records and shadow lines, or single record view field data lines. Other lines in the display area, i.e. Header lines and blank filler lines that precede a group of header lines or follow the "End of Data" record, are not included in this calculation.

Attempting to scroll up beyond the first entry (data record, shadow line or field data line) will make the first entry the first line of the display.

Multi Record View Scrolling:

Where the cursor is positioned at an offset within a column header line, the cursor is deemed to be located at the same offset within the data record that immediately follows the group of header lines. Therefore, UP CURSOR will operate successfully when the cursor is positioned within a header line.

The first data record in the display must be preceded by all of its header lines. To accomodate this, the magnitude by which the display is scrolled upwards may be reduced with the result that it may be impossible for the target line of the UP command to occupy the last line of the current display. The display will be scrolled upwards so that the target line is still in view though not the last line in the display. In order to overcome this, the user would have to alter the display area dimensions.

For the same reason, it is possible that execution of an UP command may result in no change to the displayed data, in which case UP PAGE is executed instead.

Parameters:

`CURSOR`
`CSR`
: The data record, shadow line or field data line on which the cursor is positioned becomes the last line of the scrolled display.
If the cursor is positioned on a header line, the data record or field data line immediately following the header line becomes the last line in the display area.
If the cursor is positioned outside the display area or on the last line within the display area, then UP PAGE is executed instead.

`DATA`
: Scroll up to display one page (display window depth) less one line of data.
The first data record, shadow line or field data line in the current display area becomes the last line of the scrolled display.

`HALF`
: Scroll up half a page of data.
The data record, shadow line or field data line that is half way down the page of data in the current display area becomes the last line of the scrolled display.

`MAX`
: Scroll up to display the first page of data.
The "Top of Data" line becomes the first line of the scrolled display.
Equivalent to the TOP command.

`PAGE`
: Scroll up to display the next whole page of data.
The data record, shadow line or field data line before the first line of the current display area becomes the last line of the scrolled display.

`n_lines`
: Scroll up a specified number of lines.
The data record, shadow line or field data line that is *n_lines* lines above the current line becomes the first line of the scrolled display.

Examples:

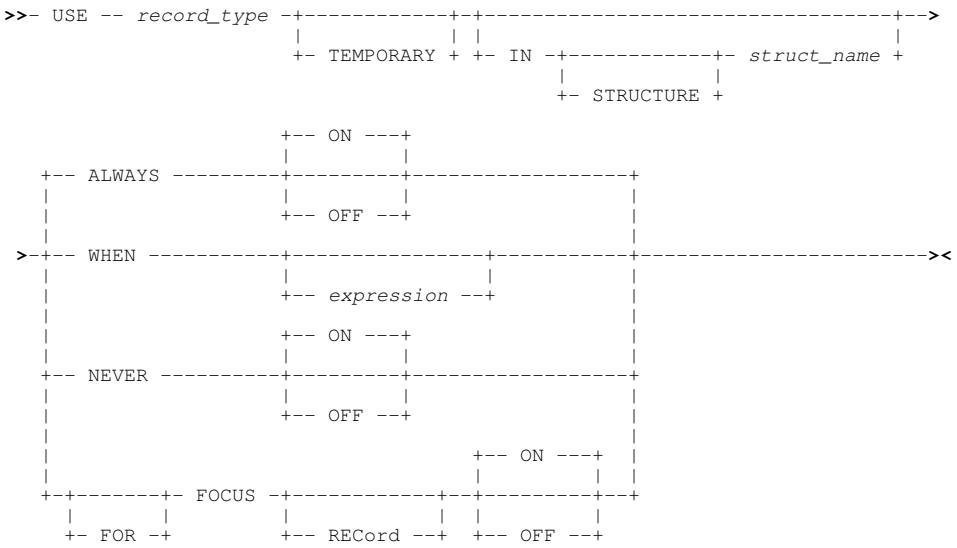
`up page`
: Scroll the display area up one page.

See Also:

DOWN
LEFT
RIGHT

USE

Syntax:



Note:

1. A user interface to the USE command (allowing the user to list and set USE WHEN/ALWAYS/NEVER conditions) is available from the SDE Edit/Browse Utility Window menu accessed using <PF4> in any SDE edit/browse session. Items 6 and 7, "Modify record-type Identification criteria" and "Override record-type for focus line" respectively, relate directly to the USE command.

2. USE does **not** affect data in the file, only the way in which the data is formatted. Unless FOR FOCUS or TEMPORARY is specified, USE may update the in-storage copy of the SDE structure (SDO) so that, when the structure is dropped, the user will be prompted to save the updated SDO to disk.

   Because no data is changed, USE is not included in the UNDO/REDO chain.

Description:

The USE command is not applicable to DB2 Table edit.

Where individual records are not automatically assigned the required record type (RTO) defined within the structure (SDO), then the **USE** command may be executed to express more sophisticated record type assignment criteria or force a specific record type assignment.
See "*Record Type Assignment*".

Incorrect record type assignment may occur if the total length of the field definitions within a record type matches that of another record type. Similarly, no record types may exist that exactly match the length of the data record, in which case the first record type in the SDO is assigned by default.

USE WHEN *expression* may be specified to one or more RTO definitions in order to sophisticate automatic assignment of record type, so matching records with their correct RTO.
USE NEVER ON will exclude an RTO definition from being eligible for assignment.
USE ALWAYS ON will force all records to be assigned a single RTO definition.
USE FOR FOCUS RECORD will force temporary assignment of a specific RTO definition to the record occupying the focus line.

Unless TEMPORARY or FOR FOCUS RECORD is specified, execution of the USE command will affect a change to the RTO record type definition, identified by *record_type*, in the specified SDO. If not already loaded, the required SDO is loaded from its structured data file (SDF) and updated accordingly. If the SDO containing this record type is a non-temporary structure, then the change may be saved to the structure definition file when the structure is dropped or on execution of the SAVESTRUCTURE command.

Any update to record type assignment criteria via the USE command, is **immediately** applied to in-storage records belonging to the current SDE window that are assigned record types from the updated structure. Except for USE FOR FOCUS RECORD, execution of the USE command will force record type assessment of **all** the in-storage records based on the standard RTO assignment precedence.

Records in an active SDE edit or browse session that use an updated structure but are not in the current SDE window, will be unaffected by the update until record type assessment is forced for that SDE window (i.e. via another USE command.) Any new SDE edit or browse session that uses an updated structure will obey the updated record selection criteria.

If either USE ALWAYS ON or USE NEVER ON are executed, then a USE WHEN *expression* defined in the RTO is preserved so that, following execution of USE ALWAYS OFF or USE NEVER OFF, records will be re-assigned their original RTOs based on the RTO defined WHEN expressions.

Parameters:

record_type
Identifies the record type to which assignment criteria will be applied.

TEMPORARY
Indicates that assignment of record types based on this USE command is to be temporary. Unless further non-temporary USE commands are executed, the user will not be prompted to save these changes to the in-storage structure (SDO).

IN (STRUCTURE) *struct_name*
Identifies the SDO that contains the specified record type. Defaults to the SDO of the current SDE window.

ALWAYS ON|OFF
ALWAYS ON forces the specified record type to be assigned to all records that use the specified structure so overriding any other record type assignment criteria. If the specified record type has been defined with USE NEVER ON, then USE NEVER OFF is set for that record type. Note that USE ALWAYS ON will not remove any USE WHEN *expression* defined to the record type.

ALWAYS OFF removes the ALWAYS ON definition for the specified record type and so all records that use the specified structure are assigned a record type based on other criteria.

ALWAYS ON is default if USE is executed without parameter ALWAYS, WHEN, NEVER or FOR FOCUS RECORD. Similarly, ALWAYS ON is default if ALWAYS is specified without ON or OFF.

WHEN *expression*
Specifies the criteria that must be satisfied before the record type can be assigned to a particular record.

expression is a valid SDE expression which supports function calls, *record_type* field names and references, sub-expressions, arithmetic, relational and logical operators. The result of the WHEN expression must be numeric and is treated as being Boolean in nature with a zero value indicating a "false" condition and any non-zero value indicating a "true" condition.

If a WHEN expression returns a "true" result for a data record, then the record type to which it is defined is assigned to that record.

If WHEN is specified with no *expression*, then any existing WHEN *expression* will be removed from the record type definition.

NEVER ON|OFF
NEVER ON indicates that the specified record type should never be assigned to a record and so this record type is removed from record type assignment processing. If the specified record type has been assigned to all records with USE ALWAYS ON, then USE ALWAYS OFF is set. Note that USE NEVER ON will not remove any USE WHEN *expression* defined to the record type.

NEVER OFF removes the NEVER ON definition for the specified record type and so reinstates the record type as being eligible for record type assignment processing.

NEVER ON is default if NEVER is specified without ON or OFF.

(FOR) FOCUS (RECORD) < ON|OFF >
FOR FOCUS RECORD assigns the specified record type to the record occupying the focus line only. If the specified record type has been defined with USE NEVER ON, then this flag is temporarily ignored for the focus line record only. Note that USE FOR FOCUS RECORD will not remove any USE WHEN *expression* or USE NEVER ON flag defined to the record type.

FOR FOCUS RECORD OFF removes the record type assignment for the record occupying the focus line and subsequently performs standard record type assignment processing for that record only.

Examples:

```
use   Type1Rec  when  col1 = 'A'  and  (col2 << 'CDE' or col3 > 42)
```
Define USE WHEN record type assignment criteria for record type Type1Rec in the SDO used by data records in the current SDE window. This *expression* would match records for which the value in field COL1 is equal to 'A' and either of the following are true:
- ◊ Value in field COL2 contains the literal string 'CDE'. (i.e. 'CDE' in any character case).
- ◊ Value in field COL3 is numeric and greater than 42.

```
use   Type1Rec  always
```
Assign record type Type1Rec to all records.

```
use   Type1Rec  never
```
Never allow record type Type1Rec to be assigned to a record. If executed following the previous USE ALWAYS example, then ALWAYS OFF is set and record type assignment processing is performed without Type1Rec.

See Also:

DROP
RCOLOUR
SAVESTRUCTURE

VFMT

Syntax:

```
>>-- VFMT ------------------------------------------------------------------><
```

Description:

Display records or record segments in multi-record, formatted view.

If the record-type (RTO) assigned to each record or record segment has been disabled (record data is unformatted), then VFMT will re-enable the assigned record-types to reformat the record data.

Compare with FORMAT TABLE which does not attempt to re-enable the assigned record-types.

Parameters:

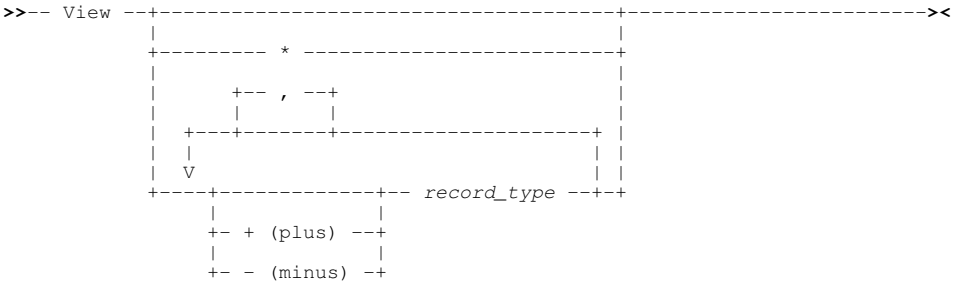
VFMT has no parameters.

See Also:

FORMAT
CHAR
MAP (FMT)
UNFMT
ZOOM

VIEW

Syntax:



Description:

Select the record types associated with data records to be made visible within an SDE BROWSE or EDIT window. The VIEW command performs the same operation as the "V" prefix area command.

Records that not included in the display of records due to a VIEW CLI command or "V" prefix command, are flagged as being SUPPRESSED. Suppressed records can be identified if shadow lines are set on for SUPPRESSED records.

Where a non-generic record type is specified, the DRECTYPE setting is updated to be the first *record_type* parameter specified on the VIEW command.

If no parameters are specified, then the default record type is used.

Where more than one record type is selected in a multi record view, the appropriate column headings are displayed at each change of record type.

In a single-record view, horizontal scrolling occurs beteween visible records only.

VIEW is assigned to **PF4** by default.

Parameters:

`+ (plus)`
: Add records of the specified record type to the display of existing visible record types.

`- (minus)`
: Remove records of the specified record type from the display of existing visible record types.

record_type
: The record type of any RTO defined within the current structure definition object (SDO) or the internal record type, "Record" (or "UnMapped").
If no "+" (plus) or "-" (minus) qualifier is specified, then all other record types will be suppressed unless they are also named as parameters on the same VIEW command.
If no *record_type* is specified, the default record type is used.

`* (asterisk)`
: Generic record type indicating that records of all record types within the SDO are to be made visible including the internal record type "Record" ("UnMapped") which is used to map data records that have no associated RTO within the SDO. If supplied, this must be the only parameter.

Examples:

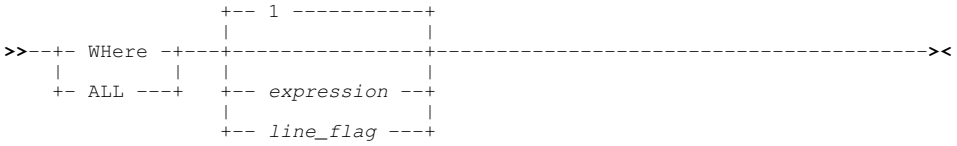
`view CompUnit,Source,Instruction`
: View records of the specified record types only. All other records will be suppressed.

`view *`
: View records of all record types.

`v +REC_CARD, -REC_CUST`
: Add records of record type REC_CARD to the current display of records and suppress records of record type REC_CUST.

WHERE

Syntax:



Description:

Display only those records (data set records, record segments or DB2 table rows) that satisfy the specified *expression* or *line__flag* criteria.

The WHERE operation tests each record in turn (including excluded records), starting at the first record selected for EDIT/BROWSE. Records that satisfy the specified criteria are included for display whereas records that do not satisfy the specified criteria are excluded.

The first record that satisfies the specified WHERE criteria becomes the current line, otherwise, if no records satisfy the criteria, the Top of Data line becomes the current line.

If *expression* is specified which includes reference to one or more formatted field names or field reference numbers, then WHERE will apply only to records that are of the default record type. This record type also becomes the new value of the DRECTYPE option. Records not assigned the default record type will not be tested and so are unaffected by this type of WHERE operation. Note, that VIEW may be used to suppress display of records assigned other record types, if required.

If WHERE is executed with no parameters, then all non-suppressed records are included in the display. To include only records assigned a specific record type (e.g. the default record type after execution of WHERE with an expression involving references to formatted fields), execute VIEW to suppress all other record types before executing WHERE with no parameters.

In single record view, the display area will be scrolled to the first data record that satisfies the specified criteria.

Note that, while WHERE filters records that have already been loaded by SDE edit/browse, the FILTER parameter of SDE EDIT and BROWSE operations will filter records before being loaded into storage.

Parameters:

expression

expression is a valid SDE expression which supports function calls, *record_type* field names and references, sub-expressions, arithmetic, relational and logical operators. The result of *expression* must be numeric and is treated as being Boolean in nature with a zero value indicating a "false" condition and any non-zero value indicating a "true" condition.

If the result of *expression* is "true" then the record being tested is selected for display.

Default *expression* is "1" indicating that all non-suppressed records are selected for display.

line_flag

line_flag may be specified to locate a record or record segment that has been flagged with the requested line flag.

Each valid *line_flag* keyword detailed below corresponds to a built-in function.

A description of each *line_flag* keyword may be found in its equivalent function description.

***line_flag* Keyword**	**Built-in Function**
ERRor	CHANGEERROR()
CHAnge | Chg	CHANGEOK()
ALTered | UPDated	CHANGED()
DATaerror	DATATYPEERROR()
DUPkey	DUPLICATEKEY()
EMPTY	EMPTYSLOT()
EXcluded | X	EXCLUDED()
LABel	HASPOINT()
COMmand | CMd	HASPREFIXCMD()
IDentify | IDrequired	IDREQUIRED()
NEW	INSERTED()
KEYChanged | KEYChg	KEYCHANGED()
LENgtherror	LENGTHERROR()
EOL | NOEOL	NOEOL()
SAVE	SAVEREQUIRED()
TRNC | TRUNCated	TRUNCATED()
VALERRor	VALUEERROR()

Examples:

`where LastName = C'Jones'`

For records that are of the default record type, display only those records for which the field "LastName" is equal to "Jones" in mixed case.

`where LastName = C'Jones' & Salary > 26000`

As above but filter the data records further so that the numeric field "Salary" is greater than 26,000.

`where #12(2,7) \<< 'E77'`

Display records that are of the default record type and where the two dimensional character array element belonging to field reference number 12 does not contain the string "E77".

`where PostCode >> "SW1" & ( DispatchDate = OrderDate | ( Quantity > 30 & Cost < 2155.53 ) )`

Display records that are of the default record type where character field "PostCode" begins with string "SW1" **AND** one of the following conditions are true:

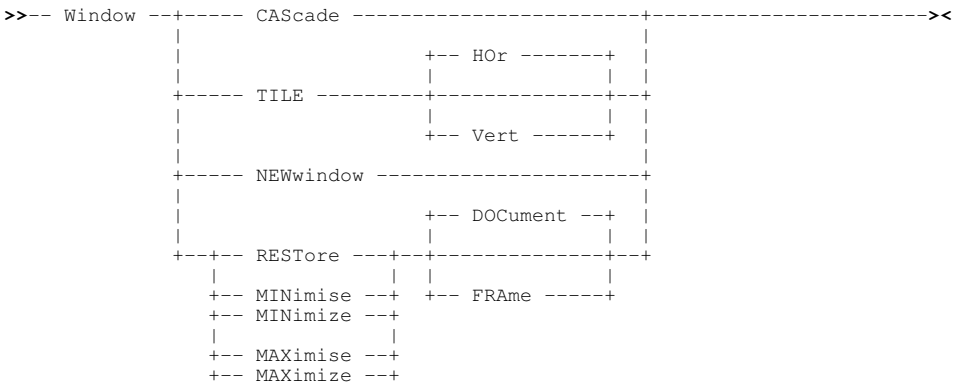
1. The contents of fields "DispatchDate" and "OrderDate" are equal.
2. The numeric field "Quantity" is greater than 30 **AND** the numeric field "Cost" is less than 2155.53.

See Also:

FLIP
LESS
LOCATE
MORE

WINDOW

Syntax:



Description:

Perform window focusing, positioning and sizing operations on the current SDE (document) window view or CBLe MDI parent (frame) window.

Parameters:

`CASCADE`
: Cascade all MDI child windows within the CBLe MDI parent window.

`TILE`
: Tile all MDI child windows within the CBLe MDI parent window horizontally (HOR), the default, or vertically (VERT).

`NEWWINDOW`
: Open an new SDE window view for the data in the current SDE window view.

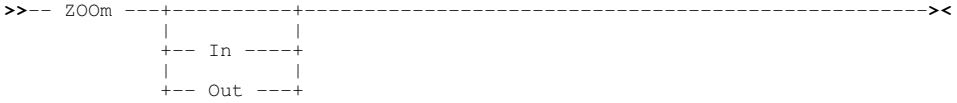
`MAXIMISE`
`MAXIMIZE`
: Maximise the current SDE window view (DOCUMENT), the default, or the CBLe MDI parent (FRAME) window.

`MINIMISE`
`MINIMIZE`
: Minimise the current SDE window view (DOCUMENT), the default, or the CBLe MDI parent (FRAME) window.

`RESTORE`
: Restore the current SDE window view (DOCUMENT), the default, or the CBLe MDI parent (FRAME) window back to its original state, prior to being maximised or minimised.

ZOOM

Syntax:



Description:

Switches the display format between single record view and multi record view. When switching to single record view the ZOOM command operates on the default record.

If no parameter is specified, then the display format toggles between the single and multi record view.

Parameters:

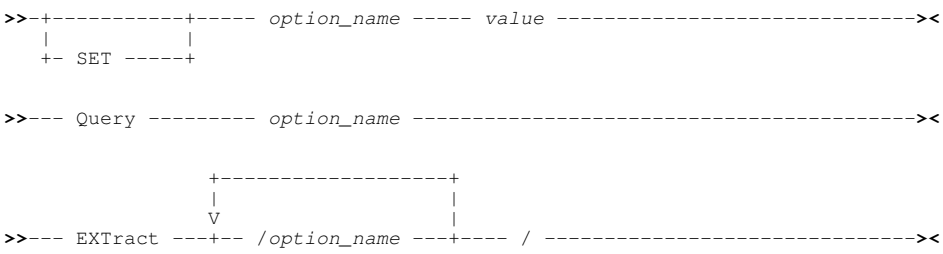
`IN`
: Switch to single record view mode. With FORMAT TABLE in effect this is equivalent to issuing the command FORMAT SINGLE.

OUT

Switch to multi record view mode. With FORMAT SINGLE in effect this is equivalent to issuing the command FORMAT TABLE.

SET/QUERY/EXTRACT

Syntax:



Description:

Structured Data Environment options may set, and their current values queried or extracted into stem-variables for use in REXX macros using the SET, QUERY and EXTRACT commands repectively. Additional operations supported by EXTRACT, allow the user to extract information about individual records and also extract record text.

SDE options may be initialised via the following methods:

1. The (SData) section of the INI file which is processed at SELCOPY/i startup.
2. The SDEPROF macro which is executed at the start of every SDE session.
3. The SDE CLI command SET, executed from the command line or an SDE macro.
4. The "Edit Options" entry of the "Options" drop down menu on the parent MDI window menu bar.

SET options take effect at the following different levels:

Global	The option affects all SDE edited files.
File	The option may be set differently for each SDE edited file.
View	The option may be set differently for each SDE window view of the same file.

Parameters:

option_name
: The SDE environment option(s). For EXTRACT, multiple options maybe requested at once by separating each with a blank or "/" (forward slash).

value
: For SET, the new value to be assigned for *option_name*.

Supported Options:

ABBREVIATION	FMODE	MAXPL1RC	RESERVED
ALT	FNAME	MAXSTOR	SCALE
AUTOSAVE	FOCUS	MBR	SESSION
AUXDSNPREFIX	FORMAT	MSGLINE	SHADOW
CAPS	FPATH	MSGMODE	SIZE
COLATTRIBUTES	FTYPE	MULTIPOINT	TITLE
COLOUR, COLOR	FVALUE	OFFSET	TYPE
COLWIDTH	IDSCOPE	PAD	UNDOING
COMPILER	IDWARNING	PFKEY	UNNAMED
DESCRIPTION	KEY	POINT	USERNAME
DRECTYPE	LASTMSG	PREFIX	USING
DSN	LENGTH	QSEPARATOR	VALUE
DSORG	LOADWARNING	RECFM	VIEW
EOLIN	LRECL	RECINFO	WINNAME
EOLOUT	MACROPATH	RECTYPES	WINPOS
FIELD	MAPPING	REFERENCE	WINSIZE
FILEID	MAXCOBOLRC	REGION	WRAP

Examples:

```
< sd query shad
< sd shad off all
< 'extract /reference/scale/type/'
```

ABBREVIATION - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- ABBREViation ----+-- ON ---+------------------------------><
   |           |                   |         |
   +- SET -----+                   +-- OFF --+

>>--- Query ------ ABBREViation ------------------------------------------------><

>>--- EXTract --- /ABBREViation/ ------------------------------------------------><
```

Description:

This option controls whether the group name and field item designators that constitute a field name may be abbreviated, starting with the first letter of the designator, in SDE commands that reference record fields by name. (e.g. FIND, CHANGE, WHERE)

For example, in formatted records that contain the field name 'SPROCKET-SIZE' in a group field 'MARCX', the field may be referenced as M.SPR if this name is unique to that field.

Note that abbreviated field names may **always** be specified for a LOCATE command, regardless of the setting of the ABBREVIATION option.

SET ABBREVIATION takes effect at the Global level.

SET Value:

`ON|OFF`
: Controls whether use of abbreviated field names is set ON or OFF.

QUERY Response:

The current setting of the ABBREVIATION option.

EXTRACT Rexx variables:

`abbreviation.0`	1
`abbreviation.1`	The current setting of the ABBREVIATION option, **ON** or **OFF**.

ALT - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option ALT, controls the same options in an SDE window view as those supported by a CBLe edit view. See SET ALT in CBLe Text Edit documentation.

AUTOSAVE - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- AUTOSave ----+-- ON ---+---+------------------------------><
   |           |                |         |   |              |
   +- SET -----+                +-- OFF --+   +-- PROMPT ----+
                                              |              |
                                              +-- NOPROMPT --+

>>--- Query ------ AUTOSave ------------------------------------------------><

>>--- EXTract --- /AUTOSave/ ------------------------------------------------><
```

Description:

This option controls the action taken if unsaved changes exist in edited data and the END (or QUIT) command is executed. Actions are as follow:

AUTOSAVE	Action Taken
ON (PROMPT or NOPROMPT)	Changes to the data are saved without prompting the user before doing so.
OFF NOPROMPT	Changes to the data are discarded. (Same as QQUIT)
OFF PROMPT	User is prompted to save the changes, discard the changes or to terminate the END command and return to the SDE edit view.

The initial value of AUTOSAVE is set by the INI variable SDE.AutoSave or, where this INI option has not been set, defaults to AUTOSAVE OFF PROMPT.
When SET AUTOSAVE is executed, the SDE.AutoSave option is set or updated in the SELCOPY/i User INI file. when the SELCOPY/i session is closed. This means that the auxiliary data set name will be set across SELCOPY/i sessions.

SET AUTOSAVE takes effect at the Global level.

SET Value:

`ON|OFF`
: Controls whether changes to data will be automatically saved (ON) or not (OFF).

`PROMPT|NOPROMPT`
: For AUTOSAVE OFF only, controls whether a prompt window is opened containing choices to save the changes, discard them or cancel END and return to the edited data.

QUERY Response:

The current setting of the AUTOSAVE option.

EXTRACT Rexx variables:

`autosave.0`	2
`autosave.1`	The current setting of automatic SAVE, **ON** or **OFF**.
`autosave.2`	The current setting of the prompt window, **PROMPT** or **NOPROMPT**.

AUXDSNPREFIX - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+------------+-- AUXdsnprefix --- dsn_prefix ------------------------------><
   |            |
   +- SET -----+

>>--- Query ------ AUXdsnprefix ----------------------------------------------><

>>--- EXTract --- /AUXdsnprefix/ ----------------------------------------------><
```

Description:

This option controls the default data set name HLQ prefix used to allocate the auxiliary data set when the Auxiliary Edit technique is required.

The initial value of AUXDSNPREFIX is set by the INI variable SDE.AuxDSNPrefix or, where this INI option has not been set, defaults to %USER%.CBLI.SDEAUX.
When SET AUXDSNPREFIX is executed, the SDE.AuxDSNPrefix option is set or updated in the User INI file when the SELCOPY/i session is closed. This means that the auxiliary data set name will be set across SELCOPY/i sessions.

SET AUXDSNPREFIX takes effect at the Global level.

SET Value:

`dsn_prefix`
: Valid data set name high level qualifiers, conforming to your system's standards, to be used as the DSN prefix for the generated auxiliary data set.

Qualifiers of the format **.Dyyyyddd.Thhmmssx**, representing the current local date and time, are appended to *dsn_prefix* to complete the auxiliary data set name. If *dsn_prefix* exceeds 26 characters, then the DSN is truncated.

QUERY Response:

The auxiliary DSN prefix that is the current setting of the AUXDSNPREFIX option.

EXTRACT Rexx variables:

auxdsnprefix.0	1
auxdsnprefix.1	The auxiliary DSN prefix that is the current setting of the AUXDSNPREFIX option.

CAPS - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+------------+-- CAPs ----+-- ON ---+-------------------------------------><
   |            |            |         |
   +- SET -----+             +-- OFF --+

>>--- Query ------ CAPs ------------------------------------------------------><

>>--- EXTract --- /CAPs/ ------------------------------------------------------><
```

Description:

This option controls whether upper casing of alpha characters will occur for all data within fields that are changed.

If record or record segment data is unformatted (record type "UnMapped" or "UnMappedSeg" respectively), then all alpha characters in the unformatted data will be upper cased.

SET CAPS takes effect at the View level.

SET Value:

ON|OFF
: Controls whether automatic upper casing is set ON or OFF.

QUERY Response:

The current setting of the CAPS option.

EXTRACT Rexx variables:

caps.0	1
caps.1	The current setting of CAPS option, **ON** or **OFF**.

COLATTRIBUTES - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+------------+-- COLAttributes ----+-- ON ---+-----------------------------><
   |            |                     |         |
   +- SET -----+                      +-- OFF --+

>>--- Query ------ COLAttributes ----------------------------------------------><

>>--- EXTract --- /COLAttributes/ ----------------------------------------------><
```

Description:

Applicable to DB2 table edit and browse only, this option controls the display of the DB2 results table column attributes when a row is presented in single format view. (See commands FORMAT and ZOOM)

Column attributes are displayed under header **PkUIFCND** and reflects information obtained from the DB2 catalog table SYSIBM.SYSCOLUMNS as follow.

Attribute	Description
Pk	A 2 byte field displaying the numeric position of the column within the Primary Key, otherwise blank.
U	Displays "u" which indicates that this column is part of a unique key, otherwise blank.
I	Displays "d" which indicates that this column is part of a non-unique (duplicate) key index, otherwise blank.
F	Displays "f" which indicates that this column is part of a referential constraint foreign key, otherwise blank.
C	Displays "c" which indicates that this column is involved in a check constraint, otherwise blank.
N	Displays "n" which indicates that this column supports a null value, otherwise blank.
D	Displays one of the following single character DEFAULT value indicators. See description of the DEFAULT column of table SYSIBM.SYSCOLUMNS in the *"IBM DB2 SQL Reference"* manual.

SET COLATTRIBUTES takes effect at the View level.

SET Value:

`ON|OFF`
: Controls whether display of DB2 column attributes is set ON or OFF.

QUERY Response:

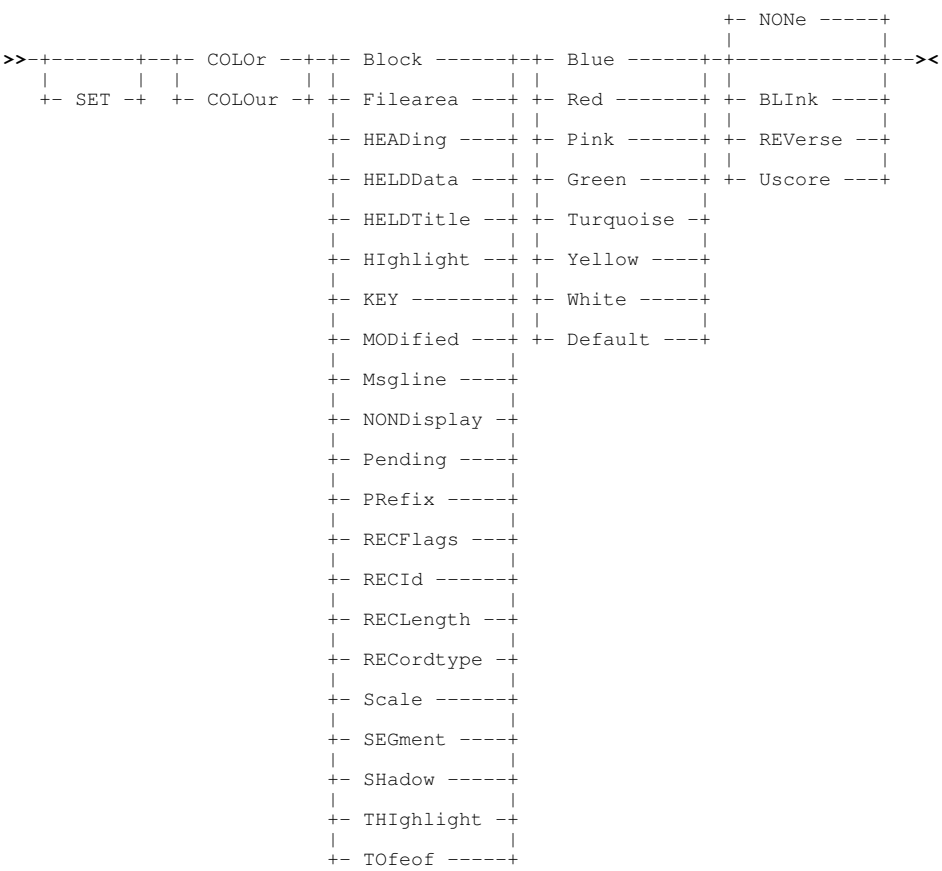
The current setting of the COLATTRIBUTES option.

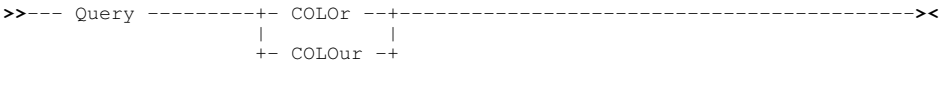
EXTRACT Rexx variables:

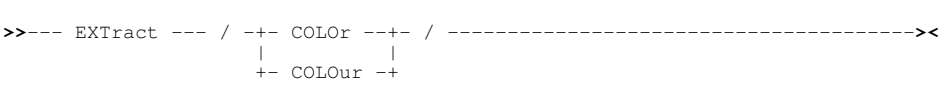
`colattributes.0`	1
`colattributes.1`	The current setting of column attributes display, **ON** or **OFF**.

COLOUR, COLOR - SET/QUERY/EXTRACT Option

Syntax:







Description:

This option controls the colour display of areas within the SDE window view.

SET COLOUR takes effect at the File level.

SET Value:

`BLOCK`
: Text within a marked block field. Default colour is BLUE REVERSE.

`FILEAREA`
: Untagged, unmarked text within display area. Default colour is BLUE

`HEADING`
: Field Name, Field Reference and Field Data Type header lines. Default colour is WHITE.

`HELDDATA`
: Field data within the display area belonging to field columns that have been HELD using the SELECT command. Default colour is TURQUOISE.

`HELDTITLE`
: Field header (and scale) lines within the display area belonging to field columns that have been HELD using the SELECT command. Default colour is YELLOW.

`HIGHLIGHT`
: Highlighted (tagged) lines. Default colour is YELLOW.

`KEY`

Data field(s) occupying the KEY field of a KSDS data set. Default colour is BLUE USCORE.

`MODIFIED`
Supported for DB2 table edit, fields containing changed data. Default colour is YELLOW.

`MSGLINE`
Message lines (with MSGLINE ON). Default colour is RED.

`NONDISPLAY`
Fields of type Character or AlphaNumeric (AN) which contain non-printable characters. Default colour is TURQUOISE USCORE.

`PENDING`
Pending commands in the prefix area. Default colour is RED.

`PREFIX`
Prefix area (with PREFIX ON). Default colour is GREEN.

`RECFLAGS`
Record Flags (with RECINFO ON FLAGS/ALL) Default colour is yellow.

`RECID`
Record TTR/OFFSET or RBA field (with RECINFO ON ID/ALL) Default colour is yellow.

`RECLENGTH`
Record Length (with RECINFO ON LENGTH/ALL) Default colour is yellow.

`RECORDTYPE`
Record type line. Default colour is PINK.

`SCALE`
Scale line (with SCALE ON). Default colour is WHITE.

`SEGMENT`
Record Segment name header line. Default colour is YELLOW.

`SHADOW`
Shadow lines (for excluded lines when SHADOW ON). Default colour is WHITE.

`THIGHLIGHT`
Highlighted targets. Default colour is GREEN REVERSE.

`TOFEOF`
Top of File and End of File lines. Default colour is YELLOW.

`BLUE | RED | PINK | GREEN | TURQUOISE | YELLOW | WHITE | DEFAULT`
Supported colours on each field. If DEFAULT is specified, the default colour for the field is set.

`BLINK | REVERSE | USCORE | NONE`
Extended highlighting of the specified field. The colour may blink, be displayed in reverse video or be underlined. Default is NONE.

QUERY Response:

For each coloured area within the SDE window display, the display area name, current colour setting and extended highlighting option is displayed on an individual message line.

EXTRACT Rexx variables:

`color.0` `colour.0`	Number of areas within the display for which a colour option may be assigned.
`color.i` `colour.i`	One stem for each area within the display. The value of each compound variable is an upper case string containing the display area name, the current colour setting and extended highlighting option.

COLWIDTH - SET/QUERY Option

Syntax:

```
>>-+-----------+-- COLWidth --- field_col --- width -------------------------><
   |           |
   +- SET -----+

>>--- Query ------ COLWidth --- field_col ---------------------------------------------><
```

Description:

This option controls the number of characters currently displayed (column width) in a specified column field.

Column width cannot be increased beyond the column's defined maximum width.

For SDE DB2 table EDIT, SDE EDIT and SDE BROWSE, the default width of a column within the display of formatted records is the maximum defined for that column. For SDE DB2 table BROWSE only, the displayed width of variable length columns is reduced to be equal to the maximum length of the data occupying a field in the column.

SET COLWIDTH values take effect at the File level.

SET Value:

field_col
: The individual column field for which the column width will be set.

  The field column may be identified by its reference number (e.g. #6) or its name (e.g. JobID).

width
: The number of characters to be set as the column width. If this value exceeds the column's maximum width, then the maximum width value is used.

QUERY Response:

The column field name referenced by *field_col* followed my the current and maximum values of the column's width.

COMPILER - SET/QUERY/EXTRACT Option

Syntax:

```
                            +-------------------------------------+
                            V                                     |
>>-+-----------+-- COMPiler ----+-- COBOL --+---- compiler_module --+---------><
   |           |                |           |
   +- SET -----+                +-- PL1 ----+

>>--- Query ------ COMPiler ------------------------------------------------><

>>--- EXTract --- /COMPiler/ -------------------------------------------><
```

Description:

This option controls the name of the COBOL and PL1 module to be called by SDE when compilation of copy books is required (for CREATE STRUCTURE.)

The initial COMPILER values are set by the INI variables SDE.COBOLCompiler and SDE.PL1Compiler or, where these INI options have not been set, defaults to IGYCRCTL for COBOL and IBMZPLI for PL1. Note that the SDE.COBOLCompiler and SDE.PL1Compiler options may be set to a fully qualified load library DSN and member name.
When SET COMPILER is executed, the SDE.COBOLCompiler and/or SDE.PL1Compiler option is set or updated in the User INI file when the SELCOPY/i session is closed. This means that the compiler module names will be set across SELCOPY/i sessions.

SET COMPILER values take effect at the Global level.

SET Value:

`COBOL`
`PL1`
Specifies that the following token is the COBOL or PL1 compiler module name.

`compiler_module`
The 8 byte member name of the compiler module.

SDE will invoke the compiler module via standard system library search chain. This replaces any fully qualified load library name that may have been included as part of the SDE.COBOLCompiler or SDE.PL1Compiler INI option specifications.

QUERY Response:

The string "COBOL", the COBOL compiler name followed by the string "PL1" and the PL1 compiler name which are the current settings of the COMPILER option.

EXTRACT Rexx variables:

`compiler.0`	2
`compiler.1`	The current setting for the COBOL compiler name.
`compiler.2`	The current setting for the PL1 compiler name.

DESCRIPTION - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+------------+-- DESCription ---- long_description ------------------------><
   |            |
   +- SET ------+

>>--- Query ------ DESCription ----------------------------------------------><

>>--- EXTract --- /DESCription/ ----------------------------------------------><
```

Description:

This option displays and assigns the long description that may be defined as part of an SDE structure (SDO) file. See CREATE STRUCTURE.

SET DESCRIPTION values take effect at the File level.

SET Value:

`long_description`
Specifies a structure description text string which may be no longer than 512 characters. Enclosing quotation marks (") or apostrophes (') may be specified and are stripped when saved in the structure.

QUERY Response:

The string "DESCRIPTION", followed by the complete structure definition text.

EXTRACT Rexx variables:

`description.0`	1
`description.1`	The complete structure definition text.

DRECTYPE - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- DRECtype --- record_type ---------------------------------><
   |           |
   +- SET -----+

>>--- Query ------ DRECtype ------------------------------------------------><

>>--- EXTract --- /DRECtype/ -------------------------------------------------><
```

Description:

This option controls the record type to be used as the default record type if SELCOPY/i SDE is unable to determine the default record type from the focus line.

The initial value of DRECTYPE is set when the SDE window is opened via an EDIT or BROWSE command. In addition to the SET DRECTYPE command, DRECTYPE may be dynamically set following a VIEW, WHERE, MORE or LESS command or any command that requires a record type.

See section *Default Record Type* for further information.

SET DRECTYPE takes effect at the View level.

SET Value:

record_type
: The record type of any RTO defined within the current structure definition object (SDO).

QUERY Response:

The current default record type followed by the record type that is the current setting of the DRECTYPE option.

EXTRACT Rexx variables:

drectype.0	2
drectype.1	The current default record type.
drectype.2	The record type that is the current setting of the DRECTYPE option.

DSN - SET/QUERY/EXTRACT Option

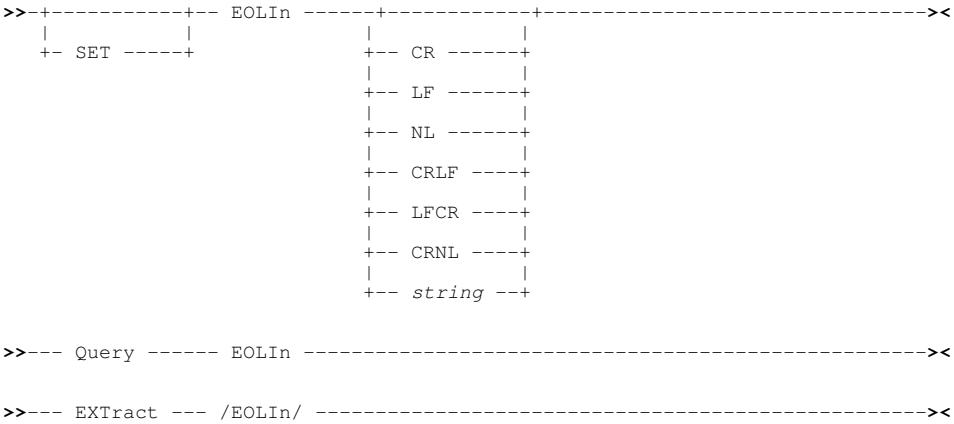
SDE SET, QUERY and EXTRACT for option DSN, has the same effect in an SDE window view as that supported by a CBLe edit view. See SET DSN in CBLe Text Edit documentation.

DSORG - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option DSORG, has the same effect in an SDE window view as that supported by a CBLe edit view. See SET DSORG in CBLe Text Edit documentation.

EOLIN - SET/QUERY/EXTRACT Option

Syntax:



Description:

EOLIN alters the current input EOL (end-of-line) delimiter string used to interpret variable length records obtained from an HFS file for SDE EDIT and BROWSE CLI commands.

An EOLIN value is set for all SDE and CBLe edit views including those containing non-HFS files.

When an edit view is opened and before the edit data is read, the default EOLIN is automatically set to be one of the following values, in the order of precedence:

1. The EOL parameter argument specified on the EDIT or BROWSE command.
2. For SDE EDIT/BROWSE only, the EOLIN value set in the SDE profile macro (using SET EOLIN).
3. The EOL format value defined in the directory entry.
4. EOLIN=NL (new line).

SET EOLIN takes effect at the File level.

SET Values:

`CR|LF|NL|CRLF|LFCR|CRNL|`*`string`*
Identifies the end-of-line delimiter. Delimiter elements are as follow:

NL	X'15'	New Line.
CR	X'0D'	Carriage Return.
LF	X'0A'	Line Feed.
string	-	A 2-byte user specified character or hex string.

QUERY Response:

The current setting of the EOLIN value.

EXTRACT Rexx variables:

`eolin.0`	1
`eolin.1`	The current setting of the EOLIN value.

EOLOUT - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- EOLOut------+------------+--------------------------------><
   |           |               |            |
   +- SET -----+               +-- CR ------+
                               |            |
                               +-- LF ------+
                               |            |
                               +-- NL ------+
                               |            |
                               +-- CRLF ----+
                               |            |
                               +-- LFCR ----+
                               |            |
                               +-- CRNL ----+
                               |            |
                               +-- string --+

>>--- Query ------ EOLOut ------------------------------------------------><

>>--- EXTract --- /EOLOut/ ------------------------------------------------><
```

Description:

EOLOUT determines the output HFS file EOL (end-of-line) delimiter string to be used when saving edited data to an HFS fileid which is not fixed format. i.e RECFM F is **not** the current setting for the edited data.

By default, the EOLOUT value is equal to the EOLIN value established when the CBLe edit view is initially opened for edit or browse. When using SDE Update-in-place Edit, EOLOUT cannot be changed and must equal the EOLIN value when the data was read.

An EOLOUT value is also set for non-HFS files allowing the user to subsequently save the data in the edit view to a new HFS file simply by using the SAVE *fileid* command where fileid is an HFS path name.

SET EOLOUT takes effect at the File level.

SET Values:

`CR|LF|NL|CRLF|LFCR|CRNL|`*`string`*

Identifies the end-of-line delimiter. Delimiter elements are as follow:

NL	X'15'	New Line.
CR	X'0D'	Carriage Return.
LF	X'0A'	Line Feed.
string	-	A 2-byte user specified character or hex string.

QUERY Response:

The current setting of the EOLOUT value.

EXTRACT Rexx variables:

`eolout.0`	1
`eolout.1`	The current setting of the EOLOUT value.

FIELD - EXTRACT Option

Syntax:

```
>>--- EXTract --- /FIELD/ ------------------------------------------------><
```

Description:

For use in macros, EXTRACT FIELD obtains the characteristics (field reference number, field data type, field name and currently displayed field width) of each field column header belonging to the default record type.

Field data type may be one of the following:

BINTEGER	FIXED	INTEGER	XVARCHAR
BIT	FLOATBIN	STRUCTURE	ZONED
CHARACTER	FLOATHEX	UNION	
DECIMAL	HEXADECIMAL	VARCHAR	

Values are obtained for each displayed field column in the order in which they appear in the display. Therefore, the SELECT command will influence the EXTRACT FIELD values.

EXTRACT Rexx variables:

`field.0`	Number of currently selected fields belonging to the default record type.
`field.i`	The blank delimitted field reference number (#nn), field data type, field name (including any array element subscripts in parentheses) and the current field width of the *i*th field in the default record type.

See Also:

QUERY/EXTRACT COLWIDTH
EXTRACT FOCUS
EXTRACT FVALUE
EXTRACT VALUE

FILEID - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option FILEID, has the same effect in an SDE window view as that supported by a CBLe edit view. See SET FILEID in CBLe Text Edit documentation.

FMODE - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option FMODE, has the same effect in an SDE window view as that supported by a CBLe edit view. See SET FMODE in CBLe Text Edit documentation.

FNAME, MBR - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option FNAME or MBR, has the same effect in an SDE window view as that supported by a CBLe edit view. See SET FNAME in CBLe Text Edit documentation.

FOCUS - EXTRACT Option

Syntax:

```
>>--- EXTract --- /FOCUS/ ------------------------------------------------><
```

Description:

For use in macros, EXTRACT FOCUS obtains information about the focus field.

EXTRACT Rexx variables:

focus.0	7
focus.1	Line number of the first record in the focus line record group.
focus.2	Line number of the last record in the focus line record group.
focus.3	Shadow state of the focus line. (VISIBLE, EXCLUDED, SUPPRESSED or NOTSELECTED)
focus.4	Description of the focus line record group. (DATA, TOF, EOF, SHADOW or BOUNDS)
focus.5	Record type of the focus line record group.
focus.6	Focus field name.
focus.7	Position of cursor within the focus field.

See Also:

EXTRACT FIELD
EXTRACT FVALUE
EXTRACT VALUE

FORMAT - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- FORmat ------+-- Character ------+------------------------><
   |           |                |                    |
   +- SET -----+                +-- Hex -------------+
                                |                    |
                                +-- Single ----------+
                                +-- Sngl ------------+
                                |                    |
                                +-- Table -----------+


>>--- Query ------ FORmat -----------------------------------------------------><


>>--- EXTract --- /FORmat/ ------------------------------------------------><
```

Description:

This option controls the display format for the current SDE BROWSE or EDIT window. SET FORMAT is equivalent to the FORMAT command.

SET FORMAT takes effect at the View level.

SET Value:

`Character`
: Multi record view with all records (or record segments) mapped as a single, character field with field name "UnMapped" (or "UnMappedSeg"). No data conversion is performed.

`Hex`
: Same as CHARACTER with the addition that the data is also displayed in Hex below the character display. Note that the Hex display occupies an additional 2 lines of data.

`Single`
`Sngl`
: Single record format with data types formatted according to the record structure. Each field occupies a separate line. Vertical scrolling transfers focus between fields. Horizontal scrolling transfers focus between records.

By default, **PF2** is assigned to distributed CBLe edit macro SDEZOOMW which opens a new view of the record data and executes FORMAT SINGLE to display the record occupying the focus line in single format.

`Table`
Multi-record table format (Default) with data types formatted according to the record structure. Each field occupies a separate column. Vertical scrolling transfers focus between records. Horizontal scrolling transfers focus between fields.

QUERY Response:

The string "FORMAT" followed by the current setting of the FORMAT option (CHAR, HEX, SINGLE or TABLE).

EXTRACT Rexx variables:

`format.0`	1
`format.1`	The current setting of the FORMAT option.

FPATH - SET/QUERY/EXTRACT Option

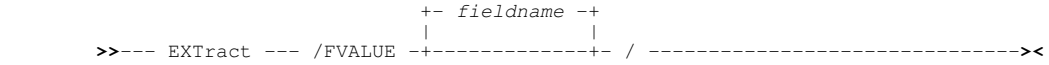
SDE SET, QUERY and EXTRACT for option FPATH, has the same effect in an SDE window view as that supported by a CBLe edit view. See SET FPATH in CBLe Text Edit documentation.

FTYPE - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option FTYPE, has the same effect in an SDE window view as that supported by a CBLe edit view. See SET FTYPE in CBLe Text Edit documentation.

FVALUE - EXTRACT Option

Syntax:



Description:

For use in macros, EXTRACT FVALUE generates a REXX variable with an assigned value, one for each field column selected for display and belonging to the default record type. Unnamed field columns are ignored.

Alternatively, if a specific field name (*fieldname*) is specified then a REXX variable with assigned value is generated for this field only. Note that *fieldname* must also have been selected for display and must belong to the default record type.

The generated REXX variable names are the record field column names, and their assigned values are the character representation of the individual field's contents within the record occupying the focus line. No variables are generated if the focus line is an EXCLUDED, SUPPRESSED or NOT SELECTED shadow line.

Field names that are elements of an array, and so have array suffices, generate REXX compound variables where the field name is the variable name stem and each dimension of the array is represented within the variable name tail. e.g. Field name "AddrLine(6)" generates variable "AddrLine.6".

Any occurrence of the "-" (minus) character in a field column name (as supported by COBOL) is translated to "_" (underscore) in the generated variable name thus avoiding REXX "Bad Arithmetic Conversion" errors. e.g. Field name "XX-HRMN" generates variable "XX_HRMN".

Beware of using variables names within your REXX macro procedure that match field column names in your structured record data. REXX variable values generated by EXTRACT FVALUE will replace any existing value already assigned to a variable name that matches a record field column name.

Because only variables for displayed field columns of the default record type are generated, the SELECT command will influence the effect of EXTRACT FVALUE.

EXTRACT Rexx variables:

EXTRACT FVALUE assigns variables as described above. No compound variable with a stem of "fvalue" is assigned.

See Also:

EXTRACT FIELD
EXTRACT FOCUS
EXTRACT VALUE

IDSCOPE - SET/QUERY/EXTRACT Option

Syntax:

```
                              +-- ALL -------+
                              +-- CHANGED ---+
                              |              |
>>-+-----------+-- IDScope -----+--------------+-----------------------------><
   |           |                |              |
   +- SET -----+                +-- FLAGGED ---+

>>--- Query ------ IDScope ------------------------------------------------><

>>--- EXTract --- /IDScope/ -----------------------------------------------><
```

Description:

This option controls which record flag must be set on in order for that record to be selected for remap by the IDENTIFY command.

Note that, for segmented record edit, a record may be selected for remap if the required flag is set on for any of its primary or secondary segments.

SET IDSCOPE takes effect at the View level.

SET Value:

`ALL|CHANGED`
: Records may be selected for remap if either the ID flag or CHANGED flag is set on.
Default is ALL which is a synonym for CHANGED.

`FLAGGED`
: Records may be selected for remap only if the ID flag is set on.

QUERY Response:

The current setting of the IDSCOPE option (CHANGED or FLAGGED).

EXTRACT Rexx variables:

`idscope.0`	1
`idscope.1`	The current setting of IDSCOPE, **CHANGED** or **FLAGGED**.

IDWARNING - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- IDWarning ----+-- ON ---+--------------------------------><
   |           |                 |         |
   +- SET -----+                 +-- OFF --+

>>--- Query ------ IDWarning ------------------------------------------------><

>>--- EXTract --- /IDWarning/ ---------------------------------------------><
```

Description:

This option controls whether the prefix area ID warning symbol **==ID>** is displayed when data in an ID field, identified by USE WHEN criteria, is changed in any way.

The ID warning symbol indicates that the IDENTIFY command should be executed to possibly remap data in the record using a different record type definition.

SET IDWARNING takes effect at the View level.

SET Value:

`ON|OFF`
: Controls whether the ID warning symbol is to be displayed.

QUERY Response:

The current setting of the IDWARNING option.

EXTRACT Rexx variables:

`idwarning.0`	1
`idwarning.1`	The current setting of IDWARNING option, **ON** or **OFF**.

KEY - QUERY/EXTRACT Option

Syntax:

```
>>--- Query ------ KEY -------------------------------------------------------><

>>--- EXTract --- /KEY/ -----------------------------------------------------><
```

Description:

QUERY and EXTRACT KEY reports the data occupying the key field of the KSDS record occupying focus line of the displayed data.

QUERY Response:

The string "KEY" followed by the key field data in printable character and hexadecimal string representation.

EXTRACT Rexx variables:

`key.0`	1
`key.1`	The data in the key field of the record occupying the focus line.

LASTMSG - QUERY/EXTRACT Option

Syntax:

```
>>--- Query ------ LASTmsg ------------------------------------------------><

>>--- EXTract --- /LASTmsg/ ----------------------------------------------><
```

Description:

Obtain the last message or error message generated for the current SDE window view.

Unlike QUERY/EXTRACT LASTMSG for CBLe window views, the recalled message string is automatically returned as is. i.e. No message number and "LASTMSG" prefix string.

QUERY Response:

Displays the last message or error message generated for the current window view. The message may not have been displayed if NOMSG was used or MSGMODE was OFF.

EXTRACT Rexx variables:

lastmsg.0	1
lastmsg.1	The last message issued.

See Also:

EXTRACT MSGMODE

LENGTH - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- LENgth ------ n_bytes ------------------------------------------><
   |           |
   +- SET -----+

>>--- Query ------ LENgth ----------------------------------------------><

>>--- EXTract --- /LENgth/ ----------------------------------------------><
```

Description:

This option controls the record length of the record occupying the focus line.

If the length of a record is increased, then pad characters are appended to the record. The pad character used is that defined by the current value for PAD. Reducing a record's length will truncate record data.

The length of a record may also be updated by overtyping the new value in the Length field displayed in the Record Information area. (See SET/QUERY/EXTRACT RECINFO and RECLEN commands.)

If a structure has been applied to the edited records, then the record type (RTO) associated with the updated record is re-evaluated and the contents of the SDE edit view updated accordingly.

Record length changes are only applicable to variable length records where Full Edit, KSDS Edit or Auxiliary Edit is in effect.

Beware that changing record lengths incorrectly may result in length errors, indicated by =LGTH> or =ERRV> in the prefix area, and so the record data is no longer correctly mapped by the defined record structure.

SET LENGTH takes effect at the File level.

SET Value:

n_bytes
: A positive integer value representing the new length assigned to the record.

QUERY Response:

The length of the record occupying the focus line.

EXTRACT Rexx variables:

length.0	1
length.1	Length of the record occupying the focus line.

LOADWARNING - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+------------+-- LOADWarning --- n_recs ----------------------------------><
   |            |
   +- SET -----+

>>--- Query ------ LOADWarning ------------------------------------------------><

>>--- EXTract --- /LOADWarning/ ------------------------------------------------><
```

Description:

This option specifies the threshold of number of data set records to be loaded into storage before the Load Warning pop-up message window is displayed prompting the user to either continue or stop loading records for SDE display and edit. The message also provides the user with the option to respect or bypass the load warning when the load theshold is once again encountered on continuing to load records from the data set.

If the user chooses to interrupt load of the data set's records once the load threshold has been reached, then records that have already been loaded are displayed in the SDE window view. If the records were loaded for edit, then Update-in-place edit will be used overriding any EDIT REPLACE request for full edit capabilities.

The purpose of the Load Warning is to allow users to break out of possible time and resource consuming loads of large data sets.

Note the difference between SDE LOADWARNING and CBLe text edit LOADWARNING which specifies number of bytes loaded instead of number of records. SDE SET LOADWARNING has no affect on the CBLe text edit LOADWARNING threshold.

Load Warning applies only to edit techniques where an attempt is made to load all requested records into storage. e.g. no load warning threshold checks are made for KSDS and AUXILIARY edit techniques.

The initial LOADWARNING value is set by the INI variable SDE.LoadWarning or, where this INI option has not been set, defaults to 5000 records.
When SET LOADWARNING is executed, the SDE.LoadWarning option is set or updated in the User INI file when the SELCOPY/i session is closed. This means that the load warning threshold will be set across SELCOPY/i sessions.

SET LOADWARNING value take effect at the Global level for all SDE EDIT and BROWSE commands.

SET Value:

n_recs
: The number of records to be loaded after which the Load Warning message occurs.

QUERY Response:

The number of records to be loaded which is the current settings of the LOADWARNING option.

EXTRACT Rexx variables:

loadwarning.0	1
loadwarning.1	The current loadwarning threshold value in number of records.

LRECL - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option LRECL, has the same effect in an SDE window view as that supported by a CBLe edit view. See SET LRECL in CBLe Text Edit documentation.

MACROPATH - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option MACROPATH, has the same effect in an SDE window view as that supported by a CBLe edit view. See SET MACROPATH in CBLe Text Edit documentation.

MAPPING - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+------------+-- MAPPing ----+-- ON ---+-------------------------------------------><
   |            |               |          |
   +- SET -----+                +-- OFF --+

>>--- Query ------ MAPPing ------------------------------------------------><

>>--- EXTract --- /MAPPing/ ------------------------------------------------><
```

Description:

This option controls whether formatted data is displayed in its mapped or unmapped format.

With MAPPING OFF, the record type assigned to formatted records or record segments persists but is not used to format the display of the record data. The record data is displayed in its unformatted (unmapped) format, instead.

With MAPPING ON, the record type assigned to formatted records or record segments is re-applied to the display of the record data.

SET MAPPING does not affect unformatted record data. i.e. SDE EDIT/BROWSE of a data set no USING structure (SDO) specified.

SET MAPPING takes effect at the View level.

SET Value:

`ON | OFF`
: Formatted record data mapping is set ON or OFF.

QUERY Response:

The current setting of the MAPPING option, **ON** or **OFF**.

EXTRACT Rexx variables:

`mapping.0`	1
`mapping.1`	The current setting of the MAPPING option, **ON** or **OFF**.

MAXCOBOLRC - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- MAXCObolrc ------ value ---------------------------------><
   |           |
   +- SET -----+

>>--- Query ------ MAXCObolrc ------------------------------------------------><

>>--- EXTract --- /MAXCObolrc/  ------------------------------------------------><
```

Description:

This option controls the maximum acceptable COBOL compiler return code for which an SDE structure will be successfully generated.

The MAXCOBOLRC value is only applicable to execution of the SDE CLI command, CREATE STRUCTURE, with parameter **FROM COBOL *copybook***.

Where a COBOL return code greater than the MAXCOBOLRC value occurs, the CREATE STRUCTURE operation fails with an error message.

The initial value of MAXCOBOLRC is set by the INI variable SDE.MaxCOBOLRC or, where this INI option has not been set, defaults to 4.
When SET MAXCOBOLRC is executed, the SDE.MaxCOBOLRC option is set or updated in the User INI file when the SELCOPY/i session is closed. This means that the maximum storage value will be set across SELCOPY/i sessions.

SET MAXCOBOLRC takes effect at the Global level.

SET Value:

value
: Maximum acceptable Return Code from the COBOL compiler.

QUERY Response:

"MAXCOBOLRC" followed by a single numeric indicating the current MAXCOBOLRC value.

EXTRACT Rexx variables:

maxcobolrc.0	1
maxcobolrc.1	The value that is the current setting of the MAXCOBOLRC option.

MAXPL1RC - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- MAXPL1rc -------- value ---------------------------------><
   |           |
   +- SET -----+

>>--- Query ------ MAXPL1rc ------------------------------------------------><

>>--- EXTract --- /MAXPL1rc/ ------------------------------------------------><
```

Description:

This option controls the maximum acceptable PL/1 compiler return code for which an SDE structure will be successfully generated.

The MAXPL1RC value is only applicable to execution of the SDE CLI command, CREATE STRUCTURE, with parameter **FROM PL1 *copybook***.

Where a PL/1 return code greater than the MAXPL1RC value occurs, the CREATE STRUCTURE operation fails with an error message.

The initial value of MAXPL1RC is set by the INI variable SDE.MaxPL1RC or, where this INI option has not been set, defaults to 4.
When SET MAXPL1RC is executed, the SDE.MaxPL1RC option is set or updated in the User INI file when the SELCOPY/i session is closed. This means that the maximum storage value will be set across SELCOPY/i sessions.

SET MAXPL1RC takes effect at the Global level.

SET Value:

value
: Maximum acceptable Return Code from the PL/1 compiler.

QUERY Response:

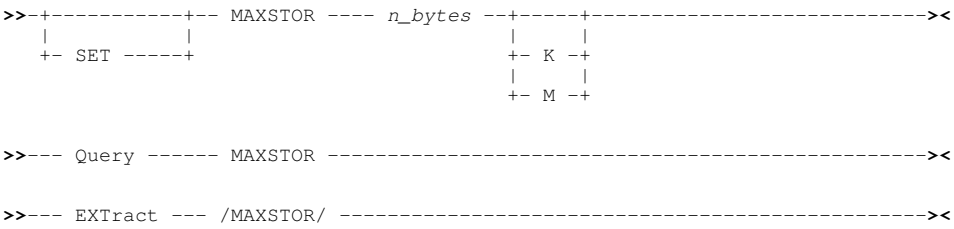
"MAXPL1RC" followed by a single numeric indicating the current MAXPL1RC value.

EXTRACT Rexx variables:

maxpl1rc.0	1
maxpl1rc.1	The value that is the current setting of the MAXPL1RC option.

MAXSTOR - SET/QUERY/EXTRACT Option

Syntax:



Description:

This option limits the amount of storage available to handle edit of any single data set within the SDE environment.

An SDE edited data set is limited by the lesser of the prevailing MAXSTOR value and the amount of free private area storage above the 16MB line available within the region at the time of open. The edited data set is unaffected by any subsequent change to the MAXSTOR setting.

This limit will be used to determine the SDE edit technique and data record management used to edit the data set. e.g. SDE EDIT of a sequential data set may be of a size which is too large to fit comfortably within the defined area of storage resulting in use of Auxiliary Edit.

If the MAXSTOR value is set to 0 (zero), it is considered to be unset and no artificial limit is imposed.

The initial value of MAXSTOR is set by the INI variable SDE.MaxStor or, where this INI option has not been set, defaults to 0 (zero).
When SET MAXSTOR is executed, the SDE.MaxStor option is set or updated in the User INI file when the SELCOPY/i session is closed. This means that the maximum storage value will be set across SELCOPY/i sessions.

SET MAXSTOR takes effect at the Global level.

SET Value:

n_bytes
*n_bytes*K
*n_bytes*M
: Amount of storage available for SDE edit of a single data set.

  Setting this value to 0 (zero) unsets the MAXSTOR limitation.

  This value may be specified as a number of bytes (*n_bytes*), number of kilobytes (*n_bytes*K) or a number of megabytes (*n_bytes*M).

QUERY Response:

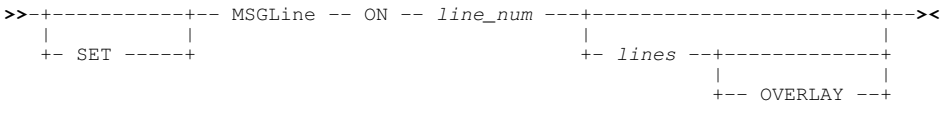
Four numeric values indicating the current MAXSTOR value and, for the current SDE edited file, the MAXSTOR value at the time the file was opened, the amount of storage allocated for the file edit so far and, of that allocation, the amount of unused storage.

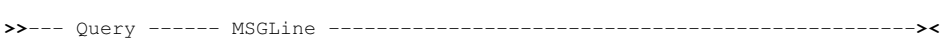
EXTRACT Rexx variables:

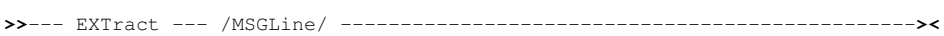
maxstor.0	4
maxstor.1	The value that is the current setting of the MAXSTOR option.
maxstor.2	The MAXSTOR value at the time the current file was opened for SDE edit.
maxstor.3	The amount of storage allocated so far for SDE edit of the current file.
maxstor.4	Of the storage allocated so far for SDE edit of the current file, the amount of storage that is unused.

MSGLINE - SET/QUERY/EXTRACT Option

Syntax:







Description:

This option controls the location and number of lines used to display messages in the SDE display window and also display properties of the first message line. The line number is specified as being relative to the top, middle or bottom of the data display area.

SET MSGLINE takes effect at the View level.

SET Value:

line_num
: Line number of first message line relative to the top, middle or bottom of the document window.

  Line numbers relative to the top of the document window are specified as positive integers. e.g. 2, 8

  Line numbers relative to the middle of the document window are specified as offsets from M. e.g. M+1, M-3

  Line numbers relative to the bottom of the document window are specified as negative integers. e.g. -6, -10

  By default *line_num* is set to 1.

lines
: The number of lines that message text may occupy within the display area.

  If the message text exceeds the number of message lines then an SDE Message List window is opened to display the message text instead.

  If *lines* is not specified, the number of message lines last defined in the current CBLe view, is unchanged.

  By default *lines* is set to 5.

OVERLAY
: Specifies that the first message line should overlay a line normally used to display a line of data. If omitted, the first message line will be reserved for message display only. Second and subsequent message lines always overlay data lines.

  Initially OVERLAY is set on.

QUERY Response:

The current setting of the MSGLINE option, **ON** or **OFF** followed by location of the first message line, the number of message lines and whether OVERLAY is in effect.

EXTRACT Rexx variables:

msgline.0	4
msgline.1	The current setting of the MSGLINE option, **ON** or **OFF**.
msgline.2	The current position of the first message line.
msgline.3	The number of message lines.
msgline.4	"OVERLAY", if overlay is in effect.

MSGMODE - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option MSGMODE, has the same effect in an SDE window view as that supported by a CBLe edit view. See SET MSGMODE in CBLe Text Edit documentation.

MULTIPOINT - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- MULTIPoint --+-- ON ---+-----------------------------------><
   |           |                |         |
   +- SET -----+                +-- OFF --+

>>--- Query ------ MULTIPoint ----------------------------------------------><

>>--- EXTract --- /MULTIPoint/ ----------------------------------------------><
```

Description:

This option controls whether or not more than one label name may be assigned to a line in the file display via the SET POINT command or by overtyping a name in the line's prefix area.

SET MULTIPOINT takes effect at the File level.

SET Value:

`ON`
: Allow multiple label names on a single line.

`OFF`
: Do not allow multiple label names on a single line. (Default)

QUERY Response:

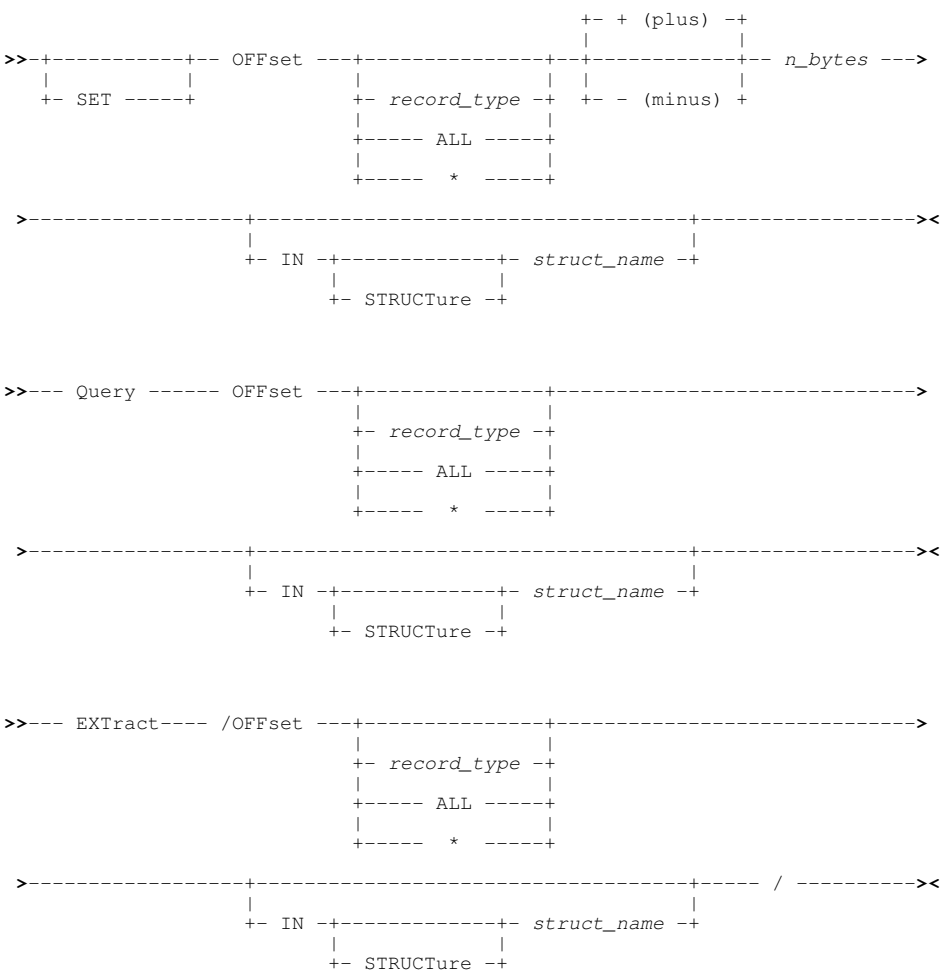
The current setting of the MULTIPOINT option, **ON** or **OFF**.

EXTRACT Rexx variables:

multipoint.0	1
multipoint.1	The current setting of the MULTIPOINT option, **ON** or **OFF**.

OFFSET - SET/QUERY/EXTRACT Option

Syntax:



Description:

This option determines the offset into a record at which data mapping by the associated record type object (RTO) is to begin. This offset may be a negative value, effectively beginning the record map before the start of the record data itself and so bypassing fields defined at the start of the RTO.

The offset value is stored as part of the RTO definition and becomes permanent when the structure is saved (e.g. using the SAVESTRUCTURE command.)

Therefore, when SDE attempts to match an RTO within the specified structure with a particular data set record, any offset value saved within the RTO is added to that RTO's determined maximum and minimum record length values. Only if the record's length falls between these adjusted minimum and maximum values (and any USE WHEN conditions have been satisfied), will the RTO become associated with the data record.

Following execution of OFFSET, all records that are currently in storage are re-assessed and assigned an appropriate RTO accordingly.

A negative offset cannot be applied if it would result in removing from the display a field mapping which defines the length of another field, defines the size of an array or is included as part of a USE WHEN expression.

Where an RTO containing a negative offset value is associated with a record, data at the beginning of the record may only be partially mapped by a field definition within the RTO. In this case, the partially mapped data is omitted so that record data is displayed starting at the first complete field mapping.
e.g. RTO begins with a 4-byte field map but has an offset value of -2, therefore data is displayed using the 2nd RTO field map starting at offset 2 into the record.

Where a positive offset is applied so that fields defined within the RTO correspond to positions beyond the length of the record data, then a length error occurs and **=ERRV>** is displayed in the record's prefix area.

SET OFFSET takes effect at the Global level.

SET, QUERY & EXTRACT Values:

record_type
: Set, Query or Extract the offset value for the specified record type (RTO).
Defaults to the Default Record Type.

ALL
*
: Set, Query or Extract the offset value for all record types (RTO) defined within the structure (SDO) *struct_name*.

+ (plus)
- (minus)
: Set a positive "+" (plus) or negative "-" (minus) offset value.
Defaults to a positive offset.

n_bytes
: Set an integer number of bytes at which record mapping is to be offset.

IN *struct_name*
: Set, Query or Extract the name of the structure (SDO) to which the specified RTO belongs.
Defaults to the SDO being used in the current SDE window.

QUERY Response:

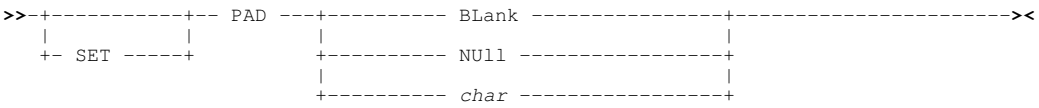
Displays the current OFFSET value for the specified record type on a single message line. If parameter ALL or * is specified, the OFFSET value for each record type within the SDO is displayed on a separate message line.

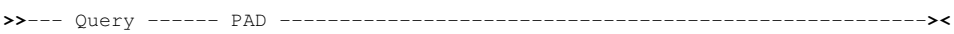
EXTRACT Rexx variables:

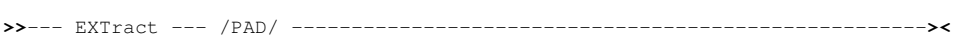
offset.0	Number of record types (RTO) if parameters ALL or * is specified; otherwise 1.
offset.i	The record type followed by its current OFFSET value.

PAD - SET/QUERY/EXTRACT Option

Syntax:







Description:

This option defines the PAD character to be used when a new record is inserted or when the length of a variable length record is increased (via a CHANGE command or by updating the Record information Length field.)

The initial value for PAD is BLANK.

SET PAD takes effect at the File level.

SET Value:

BLANK
: The blank character (x'40').

NULL
: The null character (x'00').

char
: A single byte value represented in character, hexadecimal or binary format.
Character format may be unquoted or enclosed in quotation marks (") or apostrophes (').
Hex format is a value 0-255, enclosed in apostrophes and prefixed by 'x' or 'X' (e.g. x'FE'.)
Binary format is a value 00000000-11111111, enclosed in apostrophes and prefixed by 'b' or 'B' (e.g. b'111111110'.)

QUERY Response:

The current PAD character value displayed in character (or NULL) and hexadecimal.

EXTRACT Rexx variables:

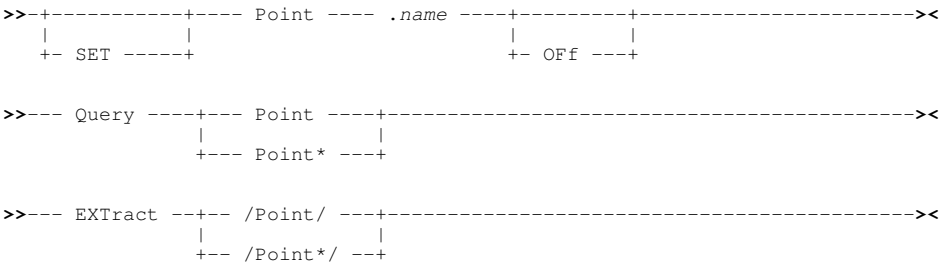
pad.0	1
pad.1	The current PAD character setting.

PFKEY - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option PFKEY, has the same effect in an SDE window view as that supported by a CBLe edit view. See SET PFKEY in CBLe Text Edit documentation.

POINT - SET/QUERY/EXTRACT Option

Syntax:



Description:

Assign or unassign a label name to the focus line (data record or shadow line) for subsequent use as a line target. The assigned label name is displayed in the line's prefix area.

If MULTIPOINT is set on, then a line may be assigned more than one label name.

The same label name may not be assigned to more than one line in the current file. Where a label name is already assigned to a line in the currunt file, it is unassigned from that line and reassigned to the focus line.

A label name remains assigned to a line even if the line record number changes due to record inserts.

SET POINT takes effect at the File level.

SET Value:

.name
: Label name of a new label to be assigned to the focus line or an existing label to be unassigned.
The preceding "." (dot) is mandatory.

OFF
: Unassign the specified label name from any of the lines in the file display.

QUERY Response:

Point
: Displays the focus line number and all label names currently allocated to the focus line. If no names are allocated to the focus line (line number n), the following message is returned:

```
ZZSD010I POINT No points assigned to line n
```

Point*
: Displays the line number and associated label names of all named lines in the current file.

EXTRACT Rexx variables:

Point

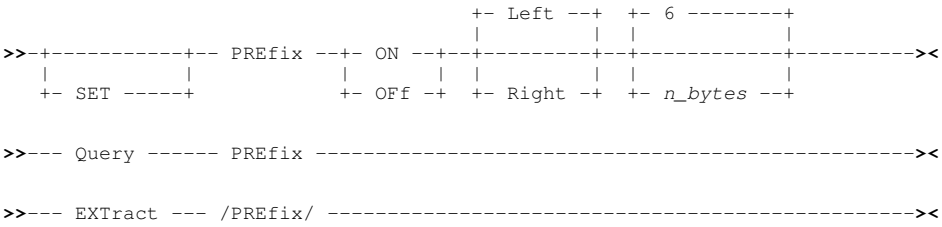
point.0	0 if focus line is not a named line; otherwise, 1.
point.1	Line number and label name(s) assigned to the focus line.

Point*

point.0	Number of named lines.
point.i	Line number and associated label name(s) of the ith named line in the current file.

PREFIX - SET/QUERY/EXTRACT Option

Syntax:



Description:

PREFIX defines whether the prefix area is displayed, whether it is displayed on the left of the window view or on the right and the number of columns to use.

The prefix displays the record number, and is also a space where line-sensitive prefix commands may be entered.

The **PREFIX** option operates at the **view-level**.

SET Value:

ON
: Display of the PREFIX area is set ON (Default).

OFF
: Display of the PREFIX area is set OFF.

LEFT
: The PREFIX area is set to display on the record's left-hand side (Default).

RIGHT
: The PREFIX area is set to display on the record's right-hand side. (i.e. it is, in fact, a suffix)

n_bytes
: The length of the prefix area. Min=1, Max=8, Default=6.

QUERY Response:

The current setting of the PREFIX option, **ON** or **OFF**, followed by **LEFT** or **RIGHT**, followed by the length **n_bytes**.

EXTRACT Rexx variables:

prefix.0	3
prefix.1	The current setting of the PREFIX option, **ON** or **OFF**.
prefix.2	The current position of the PREFIX option, **LEFT** or **RIGHT**.
prefix.3	The current length of the PREFIX area, a number from 1-8.

QSEPARATOR - QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- QSEParator --- char ------------------------------------><
   |           |
   +- SET -----+

>>--- Query ------ QSEParator ----------------------------------------------><

>>--- EXTract --- /QSEParator/ -----------------------------------------------><
```

Description:

This option controls the separator character to be used in qualified field names. By default, this is "." (dot).

The same field name may be defined in more than one nested group with in the record structure. Therefore, to reference the correct field, it should be specified using a qualified field name. e.g. *group_name.field_name*

A field's fully qualified field name includes the record type name in which it is defined in addition to the name of every group level in which it is nested. e.g. *record_type.field_name*, *record_type.group_name.field_name*. Using the record type name qualifier is required for segmented record editing where USE WHEN criteria is based on a named formatted field in a previously formatted segment.

SET QSEPARATOR takes effect at the Global level.

SET Value:

char
Specifies a field qualifier separator character.
Valid characters are: "." (dot), ":" (colon), "\" (backslash), "@" (commercial at), "^" (carat) or "%" (percent).

QUERY Response:

Displays "QSEPARATOR" followed by the qualifier separator character.

EXTRACT Rexx variables:

qseparator.0	1
qseparator.1	The qualifier separator character.

RECFM - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option RECFM, has the same effect in an SDE window view as that supported by a CBLe edit view. See SET RECFM in CBLe Text Edit documentation.

RECINFO - SET/QUERY/EXTRACT Option

Syntax:

```
                                    +-- ALL --+
                                    |         |
>>-+----------+- RECInfo --+- ON --+--+---------+---+-----------+-------------->
   |          |            |       |                 |           |
   +- SET ----+            +- OFF -+                 +- Flags ---+
                                                     |           |
                                                     +- NOFlags -+

                                                   +-- DECimal ------+
                                                   |                 |
 >--+------------+--+--------+--+-------------+---+-----------------+--------><
    |            |  |        |  |             |   |                 |
    +- Length ---+  +- Id ---+  +- SQLCode ---+   +-- HEXadecimal --+
    |            |  |        |  |             |
    +- NOLength -+  +- NOId -+  +- NOSQLCode -+
```

```
>>--- Query --- RECInfo ------------------------------------------------------><

>>--- EXTract -- /RECInfo/ --------------------------------------------------><
```

Description:

This option controls which of the standard record information columns, if any, are to be displayed for records in the current SDE window view.

For both SDE EDIT and BROWSE operations, display of record information is set off by default.

The record information columns are displayed to the left of the record data field columns. In order from left to right these are flags that have been set for the record, the length of the record and the location identification of the record within the file.

Note that, if records are of variable length and Full Edit, KSDS Edit or Auxiliary Edit is in effect, then values displayed in the record information Length column may be overtyped to update the length of a record. In all other circumstances, record information columns are non-enterable.

SET RECINFO values take effect at the View level.

SET Value:

ON
OFF
: Record information column display is switched on or off.

ALL
: Include display of all record information columns unless column display option is specified with the "NO" prefix..

FLAGS
NOFLAGS
: Include or suppress display of the record flags column.

The record flags column has a width of 4 bytes and header "Flags". Flag bytes and their meanings are, from left to right:

f	Record/segment is original, loaded from the disk copy of file.
i	Record/segment has been inserted during the edit session.
c	Record/segment has been changed since being inserted or loaded.
s	Record/segment will be saved on execution of a SAVE, or equivalent, operation.
m	Record/segment requires possible remap due to ID field change.

LENGTH
NOLENGTH
: Include or suppress display of the record length column.
This option is equivalent to the SDE command RECLEN ON/OFF.

The record length column contains a 5-byte decimal value and has header "Length". For non-DB2 table edit, the contents of this column may be overtyped to alter the length of a record if an appropriate SDE edit type is in use and the records are of variable length.

If the length of a record is increased, then pad characters are appended to the record. The pad character used is that defined by the current value for PAD. Reducing a record's length will truncate record data.

If a structure has been applied to the edited records, then the record type (RTO) associated with the updated record is re-evaluated and the contents of the SDE edit view updated accordingly.

ID
NOID
: Applicable to non-DB2 table edit only, ID or NOID respectively includes or suppresses display of the record identification column.

For non-VSAM data sets, the record identification is by Relative Volume, Track and Physical Record number (TTR) followed by an offset within the physical record. The columns are displayed as decimal or hexadecimal values with headers "Vol" (if multi-volume), "TTR" and "Offset". If the data set is single volume, the "Vol" header is unnecessary and so omitted.

For DB2 table edit, the ID column is not applicable and so is suppressed.

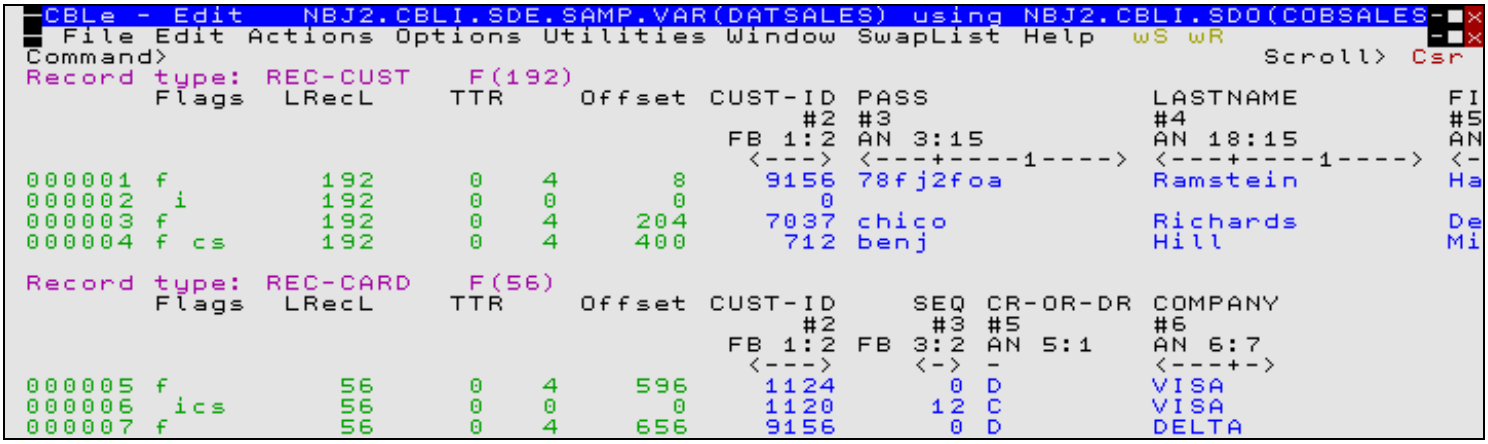

CBLe - Edit NBJ2.CBLI.SDE.SAMP.VAR(DATSALES) using NBJ2.CBLI.SDO(COBSALES
File Edit Actions Options Utilities Window SwapList Help wS wR
Command>
Scroll> Csr
Record type: REC-CUST F(192)
Flags LRecL TTR Offset CUST-ID PASS LASTNAME FI
#2 #3 #4 #5
FB 1:2 AN 3:15 AN 18:15 AN
000001 f 192 0 4 8 9156 78fj2foa Ramstein Ha
000002 i 192 0 0 0 0
000003 f 192 0 4 204 7037 chico Richards De
000004 f cs 192 0 4 400 712 benj Hill Mi
Record type: REC-CARD F(56)
Flags LRecL TTR Offset CUST-ID SEQ CR-OR-DR COMPANY
#2 #3 #5 #6
FB 1:2 FB 3:2 AN 5:1 AN 6:7
000005 f 56 0 4 596 1124 0 D VISA
000006 ics 56 0 0 0 1120 12 C VISA
000007 f 56 0 4 656 9156 0 D DELTA

Figure 41. Record Info (Decimal) for single volume non-VSAM Data Set.

For VSAM data sets, the record identification is by Relative Byte Address (RBA). The column is displayed as a decimal or hexadecimal value with header "RBA".

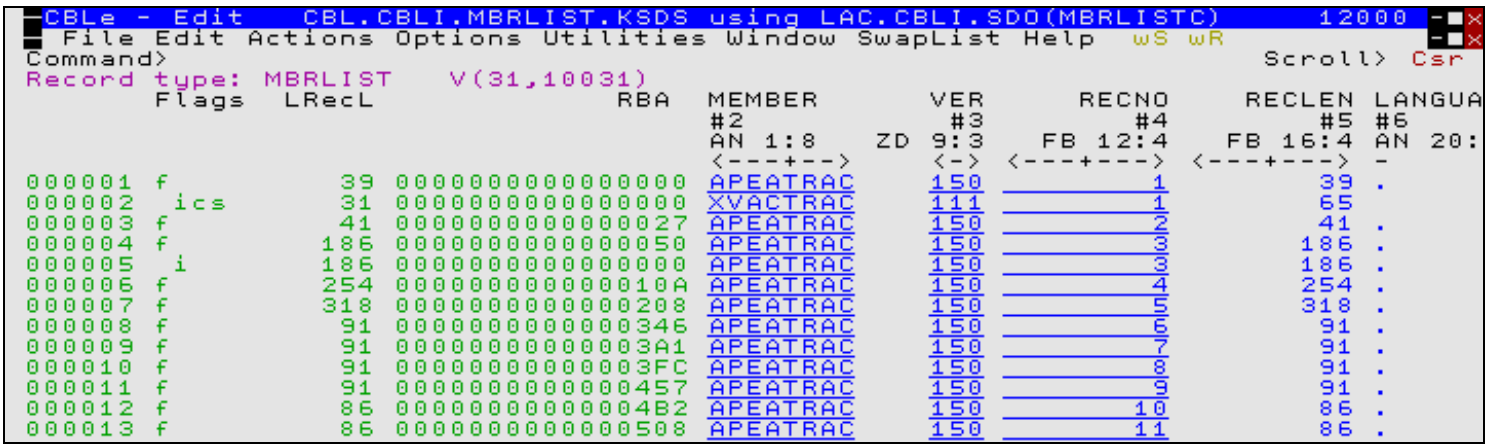


Figure 42. Record Info (Hexadecimal) for VSAM Data Set.

DECIMAL
HEXADECIMAL

Applicable to non-DB2 Table edit only, displays ID information as a decimal or hexadecimal value.

For **non-VSAM** data sets, the decimal display of Vol, TTR and Offset for the record occupies 4 columns in the following order:

1. Relative Volume number. (Displayed for multi-volume data sets only.)
2. Track number.
3. Physical record (block) number on the track.
4. Offset from the start of the physical record.

The hexadecimal display of Vol, TTR and Offset for the record is represented as 3 character fields of length 2, 6 and 8 bytes respectively, which represent a 1-byte relative volume number, 3-byte TTR and 4-byte offset hexadecimal value.

For **VSAM** data sets, the decimal display of Relative Byte Address (RBA) for the record is a single column containing a decimal value. The hexadecimal display of RBA is 16 bytes of character data representing a single 8-byte hexadecimal value.

SQLCODE
NOSQLCODE

Applicable to DB2 table edit only, SQLCODE or NOSQLCODE respectively includes or suppresses display of the SQL code column.

The SQL code column displays the last SQL code received from a an INSERT, DELETE or UPDATE statement performed on a row as a result of a SAVE operation.

For non-DB2 table edit, the SQLCODE column is not applicable and so is suppressed.

QUERY Response:

The current settings for RECINFO in order of display status ("ON" or "OFF"), column selections ("FLAGS" or "NOFLAGS" followed by "ID" or "NOID" followed by "LENGTH" or "NOLENGTH"), ID display format ("HEX" or "DEC") and SQL code display ("SQLCODE" or "NOSQLCODE").

EXTRACT Rexx variables:

`recinfo.0`	5
`recinfo.1`	The current setting of the record information column display, **ON** or **OFF**.
`recinfo.2`	The current value for selection of the record flags column, **FLAGS** or **NOFLAGS**.
`recinfo.3`	The current value for selection of the record identification column, **ID** or **NOID**.
`recinfo.4`	The current value for selection of the record length column, **LENGTH** or **NOLENGTH**.
`recinfo.5`	The current value for display of ID information, **HEX** or **DEC**.
`recinfo.6`	The current value for selection of the SQL code column, **SQLCODE** or **NOSQLCODE**.

RECTYPES - QUERY/EXTRACT Option

Syntax:

```
>>--- Query ------ RECTypes ------------------------------------------------><

>>--- EXTract --- /RECTypes/ ------------------------------------------------><
```

Description:

Obtain all record type object (RTO) names in the current SDO.

QUERY Response:

Displays all the RTO names on a single message line.

EXTRACT Rexx variables:

`rectypes.0`	Number of RTOs in the current SDO.
`rectypes.i`	The record type name of the *i*th RTO in the SDO.

REFERENCE - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- REFerence --+-- ON ---+----------------------------------><
   |           |               |         |
   +- SET -----+               +-- OFF --+

>>--- Query ------ REFerence ------------------------------------------------><

>>--- EXTract --- /REFerence/ ------------------------------------------------><
```

Description:

This option controls the display of the reference-numbers (**#nn**) occupying a row in the field column-headings for SD edit/browse.

The **REFERENCE** option operates at the **view-level**.

SET Value:

`ON`
: Reference number display is set ON.

`OFF`
: Reference number display is set OFF.

QUERY Response:

The current setting of the REFERENCE option, **ON** or **OFF**.

EXTRACT Rexx variables:

`reference.0`	1
`reference.1`	The current setting of the REFERENCE option, **ON** or **OFF**.

REGION - QUERY/EXTRACT Option

Syntax:

```
>>--- Query ------ REGion ------------------------------------------------><

>>--- EXTract --- /REGion/ ------------------------------------------------><
```

Description:

QUERY/EXTRACT REGION provides information about the private area region for the current MVS TCB.

REGION is not an option for the SET command.

QUERY Response:

Two pairs of values indicating the available private area storage region limit and high value both below and above the 16M line. e.g.

```
REGION Below 16M  Limit=9412608 High Value=9412608; Above 16M  Limit=33554432 High Value=33554432
```

These values are as found in the Virtual Storage Manager Local Data Area Control Block.

EXTRACT Rexx variables:

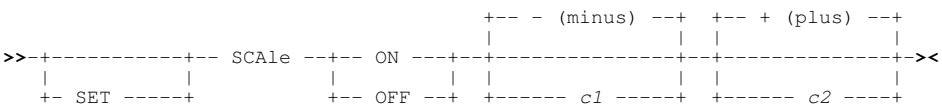
`region.0`	4
`region.1`	The below 16M line private area region limit.
`region.2`	The below 16M line private area region size (high value).
`region.3`	The above 16M line private area region limit.
`region.4`	The above 16M line private area region size (high value).

RESERVED - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option RESERVED, has the same effect in an SDE window view as that supported by a CBLe edit view. See SET RESERVED in CBLe Text Edit documentation.

SCALE - SET/QUERY/EXTRACT Option

Syntax:




```
>>--- Query ------ SCAle ------------------------------------------------><

>>--- EXTract --- /SCAle/ ------------------------------------------------><
```

Description:

This option controls the display of the scale (**<----,----1**), occupying a row in the field column-headings for SD edit/browse.

The **SCALE** option operates at the **view-level**.

SET Value:

`ON`
: Scale display is set ON.

`OFF`
: Scale display is set OFF.

c1
: The format character used for intervals of 1 byte.
Default is the minus-sign (-).
e.g. "SET SCALE ON . +" results in **'<....+....1.'**

c2
: The format character used for intervals of 5 bytes.
Default is the plus-sign (+).
e.g. "SET SCALE ON - ," results in **'<----,----1-'**

QUERY Response:

The current setting of the SCALE option, **ON** or **OFF**, followed by **c1** and **c2**.

EXTRACT Rexx variables:

`scale.0`	3
`scale.1`	The current setting of the SCALE option, **ON** or **OFF**.
`scale.2`	The current setting of the SCALE option, **c1**.
`scale.3`	The current setting of the SCALE option, **c2**.

SESSION - QUERY/EXTRACT Option

Syntax:

```
>>--- Query ------ SESSion ------------------------------------------------><

>>--- EXTract --- /SESSion/ ------------------------------------------------><
```

Description:

Obtain general information about the current Structured Data (SDE) Edit/Browse window view.

QUERY Response:

Displays SESSION followed by SDE session information.

EXTRACT Rexx variables:

`session.0`	2
`session.1`	The SDE window view type. **Browse** BROWSE. Also indicated by "Browse" in the title bar. **Reuse** EDIT REUSE. Also indicated by "Edit" in the title bar. **Update** EDIT UPDATE. Also indicated by "Edit(UP)" in the title bar. **Auxiliary** EDIT AUX. Also indicated by "Edit(AUX)" in the title bar.

session.2	The SDE record management technique. **InMemory** For EDIT REUSE only (i.e. full edit capability.), In-memory loads all records into storage and rewrites the file's records when saved. **InPlace** For EDIT UPDATE and BROWSE only, In-place keeps only displayed and modified records in storage. Record move, insert, delete and length change are not allowed. **InPlaceInMemory** For EDIT UPDATE only, InPlaceInMemory is the same as In-place except that as much of the file as possible is kept in storage. **Auxiliary** For EDIT AUX only, Auxiliary copies all records of the file to an auxiliary file in order to support full edit capability for files too large to be loaded entirely into storage (InMemory). **KSDS** For EDIT of VSAM KSDS data sets, KSDS uses the random access and update features of the VSAM KSDS organisation to provide full edit capabilities without having the whole file in storage.

SHADOW - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- SHADow --+-- ON ---+---------------------------------------->
   |           |            |         |
   +- SET -----+            +-- OFF --+

                         +-------------------- ALL ----------------------+
                         |                                               |
 >-----------------------+------------+-+---------------+-+-------------+----><
                         |            | |               | |             |
                         +- EXcluded -+ +- NOTselected -+ +- SUPressed -+

>>--- Query ------ SHADow ------------------------------------------------------><

>>--- EXTract --- /SHADow/ ---------------------------------------------------><
```

Description:

This option controls the display of shadow lines used to indicate the presence of records not displayed due to one of three reasons.

EXCLUDED identifies a record group that has been excluded by the user using selective line editing techniques e.g. using the WHERE or EXCLUDE commands.

NOTSELECTED identifies a record group that has not been assigned a record type. Reasons for this are:

1. The length of the data record does not fit that required to match any of the RTO definitions.
2. Fields within the data record do not satisfy the criteria associated with any of the available RTO definitions. (RTO criteria are defined as part of the RTO via a USE WHEN clause.)
3. Fields within the data record contain invalid data when an RTO definition is applied.

SUPPRESSED identifies a record group of a record type that has not been selected for view. (See the VIEW command.)

The **SHADOW** option operates at the **view** level.

SET Value:

ON
: Shadow display is set ON for the types that follow.

OFF
: Shadow display is set OFF for the types that follow.

ALL
: The setting affects all types of shadow lines.
SET SHADOW OFF ALL is equivalent to HIDE.
SET SHADOW ON ALL is equivalent to RESET HIDE.

EXcluded
: The setting affects shadow lines for EXCLUDED records.

NOTselected
: The setting affects shadow lines for NOTSELECTED records.

SUPressed

The setting affects shadow lines for SUPPRESSED records.

QUERY Response:

The current setting of the SHADOW options, **SHADOW EXCLUDED**, **SHADOW NOTSELECTED** and **SHADOW SUPPRESSED**, each followed by **ON** or **OFF**.

EXTRACT Rexx variables:

`shadow.0`	6
`shadow.1`	The literal **EXCLUDED**
`shadow.2`	The **ON** or **OFF** setting for EXCLUDED shadow lines.
`shadow.3`	The literal **NOTSELECTED**
`shadow.4`	The **ON** or **OFF** setting for NOTSELECTED shadow lines.
`shadow.5`	The literal **SUPPRESSED**
`shadow.6`	The **ON** or **OFF** setting for SUPPRESSED shadow lines.

SIZE - QUERY/EXTRACT Option

Syntax:

```
>>--- Query ------ SIZE ------------------------------------------------------><

>>--- EXTract --- /SIZE/ ------------------------------------------------------><
```

Description:

Obtain the number of segments and records in the current file or DB2 table.

For DB2 table rows and non-segmented records, the number of segments and records are the same.

QUERY Response:

Displays "SIZE" followed first by the number of segments then the number of records in the file or DB2 table.

EXTRACT Rexx variables:

`size.0`	2
`size.1`	The number of record segments in the file.
`size.2`	The number of records/rows in the file or DB2 table.

TITLE - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- TItle ---------- short_title ----------------------------><
   |           |
   +- SET -----+

>>--- Query ------ TItle ------------------------------------------------------><

>>--- EXTract --- /TItle/ ------------------------------------------------------><
```

Description:

This option displays and assigns the title that may be defined as part of an SDE structure (SDO) file. See CREATE STRUCTURE.

SET TITLE values take effect at the File level.

SET Value:

short_title
Specifies a structure title text string which may be no longer than 64 characters. Enclosing quotation marks (") or apostrophes (') may be specified and are stripped when saved in the structure.

QUERY Response:

The string "TITLE", followed by the complete structure title text.

EXTRACT Rexx variables:

title.0	1
title.1	The complete structure title text.

TYPE - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- TYpe --+-- ON ---+-------------------------------------><
   |           |          |         |
   +- SET -----+          +-- OFF --+

>>--- Query ------ TYpe ------------------------------------------------><

>>--- EXTract --- /TYpe/ ------------------------------------------------><
```

Description:

This option controls display of the field data type, position and length column header line for SDE EDIT and BROWSE window views.

For each field in an SDE display of file records, this column header is of the format ***data-type position*:*length*** where *position* and *length* identify the field's position and length within the unformatted record.
e.g. **'AN 111:30'** indicates an Alpha-Numeric character field of length 30 bytes starting at position 111 of the unformatted record..

For each field in an SDE display of DB2 table rows, this column header is of the format ***data-type***, ***data-type*(*length*)** or ***data-type*(*precision*,*scale*)** representing the field's DB2 built-in data type and, where applicable, its length or its precision and scale.

See *"SDE Data Types"* for descriptions of each of the supported data types and their representation in the field TYPE header line.

SET TYPE takes effect at the View level.

SET Value:

ON
OFF
Set display of the Field Type header line ON or OFF.

QUERY Response:

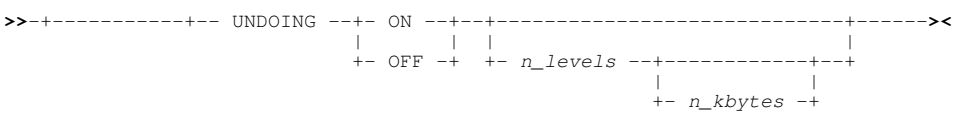
The current setting of the TYPE option, **ON** or **OFF**.

EXTRACT Rexx variables:

type.0	1
type.1	The current setting of the TYPE option, **ON** or **OFF**.

UNDOING - SET/QUERY/EXTRACT Option

Syntax:




```
>>--- Query ------ UNDOING ----------------------------------------------------><

>>--- EXTract --- /UNDOING/ -------------------------------------------------><
```

Description:

UNDOING defines whether the UNDO (and REDO) facility is enabled for SDE, the number of change levels that SDE will attempt to maintain and the maximum amount of storage SDE can allocate in order to store this information.

The third number following "Alt=" on the status line displays the current number of stored change levels.

The change level count is incremented by any command for which the UNDO operation is supported. i.e. any command that changes the data in the display area or the flag bits of a line. Flag bits include a line's excluded and suppressed indicators.

Multiple changes made to a file as a result of a macro execution are considered to be one change level only.

SDE is informed of any changes to the 3270 terminal when an Attention ID (AID) is generated (e.g. on hitting the Enter key or any of the PF Keys). It is only then that changes to the file are committed to the copy of the file data in SDE storage and the change level is updated. Therefore, where changes have been made to text on multiple lines, SDE has no indication as to the order in which the lines were changed and so assigns a change level to each updated line in ascending order of line number.

The **UNDOING** option operates at the **file-level**.

SET Value:

`ON`
`OFF`
: Set UNDOING ON or OFF.
Default is ON.

`n_levels`
: Number of change levels maintained by SDE for the file. If this value is exceeded, SDE drops the oldest undoable change level.
Default is 100.

`n_kbytes`
: Maximum amount of storage (KB) that may be obtained by SDE for storing undo information for the file.
Default is 64K.

QUERY Response:

The current setting of the UNDOING option, **ON** or **OFF** followed by number of change levels and maximum storage allocation in KBytes.

EXTRACT Rexx variables:

`undoing.0`	3
`undoing.1`	The **ON** or **OFF** setting for UNDO.
`undoing.2`	The number of change levels.
`undoing.3`	The maximum storage allocation in KBytes.

UNNAMED - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- UNNamed --+-- ON ---+-------------------------------------><
   |           |              |         |
   +- SET -----+              +-- OFF --+

>>--- Query ------ UNNamed ------------------------------------------------><

>>--- EXTract --- /UNNamed/ ------------------------------------------------><
```

Description:

This option controls whether unnamed fields appear in the display. Note that COBOL **FILLER** fields are treated as unnamed.

With UNNAMED=ON in effect unnamed fields will appear as a normal column of data in table type display, except that the field-name column heading will appear blank (or FILLER).

The **UNNAMED** option operates at the **view** level, and affects all record types.

SET Value:

ON
OFF
: Set display of UNNAMED fields ON or OFF.

QUERY Response:

The current setting of the UNNAMED option, **ON** or **OFF**.

EXTRACT Rexx variables:

unnamed.0	1
unnamed.1	The current setting of the UNNAMED option, **ON** or **OFF**.

USERNAME - QUERY/EXTRACT Option

Syntax:

```
>>--- Query ------ USERname ------------------------------------------------><

>>--- EXTract --- /USERname/ ------------------------------------------------><
```

Description:

Obtain the current user's RACF login id and the executing job name.

QUERY Response:

Displays "USERNAME RACF user" followed by the user's RACF login id, then "JOB name" followed by the currently executing job name.

EXTRACT Rexx variables:

username.0	2
username.1	The name of the user's RACF login id.
username.2	The name of the currently executing job.

USING - QUERY/EXTRACT Option

Syntax:

```
>>--- Query ------ USING ------------------------------------------------------><

>>--- EXTract --- /USING/ ----------------------------------------------------><
```

Description:

QUERY/EXTRACT USING reports the structure name (SDO) and record type of the default record within the current SDE window view.

The **USING** option operates at the **view** level.

QUERY Response:

Message ZZSD079I stating the structure name and record type. e.g.

```
ZZSD079I USING Structure CBL.CBLI.SDO(DIRAMEMP) Record type EMP
```

EXTRACT Rexx variables:

using.0	2
using.1	The structure name used in the current SDE window view.
using.2	The default record type.

VALUE - EXTRACT Option

Syntax:

```
>>--- EXTract --- /VALUE/ ----------------------------------------------------><
```

Description:

For use in macros, EXTRACT VALUE obtains the values of each field belonging to the default record type.

Values are obtained for each displayed field column in the order in which they appear in the display. Therefore, the SELECT command will influence the values returned by EXTRACT VALUE.

EXTRACT Rexx variables:

value.0	Number of currently selected fields belonging to the default record type.
value.i	The character representation of the *i*th field value in the display.

See Also:

EXTRACT FIELD
EXTRACT FOCUS
EXTRACT FVALUE

VIEW - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- View ---- view_parms ------------------------------------><
   |           |
   +- SET -----+

>>--- Query ------ View ------------------------------------------------------><

>>--- EXTract --- /View/ ------------------------------------------------------><
```

Description:

This option controls which records or record segments are in view (not suppressed) based on their assigned record types. The SET VIEW option is equivalent to the VIEW command.

SET VIEW values take effect at the View level.

SET Value:

view_parms
Specifies parameters as supported by the VIEW command.

QUERY Response:

The string "VIEW", followed by the record type names of each non-suppressed record in the file.

EXTRACT Rexx variables:

view.0	Number of non-suppressed record type names.
view.i	The name of the *i*th non-suppressed record type entry.

WINNAME - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option WINNAME, have the same effect in an SDE window view as that supported by a CBLe edit view. See SET WINNAME in CBLe Text Edit documentation.

WINPOS - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option WINPOS, have the same effect in an SDE window view as that supported by a CBLe edit view. See SET WINPOS in CBLe Text Edit documentation.

WINSIZE - SET/QUERY/EXTRACT Option

SDE SET, QUERY and EXTRACT for option WINSIZE, have the same effect in an SDE window view as that supported by a CBLe edit view. See SET WINSIZE in CBLe Text Edit documentation.

WRAP - SET/QUERY/EXTRACT Option

Syntax:

```
>>-+-----------+-- WRap ----+-- ON ---+-------------------------------------><
   |           |            |         |
   +- SET -----+            +-- OFF --+

>>--- Query ------ WRap ------------------------------------------------------><

>>--- EXTract --- /WRap/ ------------------------------------------------------><
```

Description:

WRAP defines whether a LOCATE *where_clause* search wraps around the end of the range of displayable records to continue searching until either the condition is found or the original focus line is encountered. i.e. when WRAP is set ON, searching forwards through the data continues from the Top of Data after End of Data has been reached. Likewise, searching backwards through the data continues from the End of Data when Top of Data has been reached.

Where WRAP is OFF and the End of Data or Top of Data is reached, then the following message is returned:

```
ZZSD176W Top/End of file or record range reached.
```

SET WRAP takes effect at the View level.

SET Value:

`ON`
: Allow LOCATE searches to wrap.

`OFF`
: Do not allow LOCATE searches to wrap.

QUERY Response:

The current setting of the WRAP options, **ON** or **OFF**.

EXTRACT Rexx variables:

`wrap.0`	1
`wrap.1`	The current setting of the WRAP options, **ON** or **OFF**.

Prefix Area (Line) Commands

The following commands can be entered in the prefix area of an SDE browse or edit window:

.name	Set a line pointer (line name).
A	Make this line the target for a move or copy (move or copy lines After this line).
B	Make this line the target for a move or copy (move or copy lines Before this line).
C(n) CC	Mark a line or a block of lines for copying. Lines may be copied or cut to the clipboard (using the CLIPBOARD COPY/CUT command) or copied to another position within the same edited data using prefix commands, A or B. Only visible (not excluded) data records and shadow lines representing EXCLUDED record groups are eligible to be copied. i.e. records flagged as NOTSELECTED, SUPPRESSED or, if SHADOW OFF EXCLUDED is in effect, records flagged as EXCLUDED, are **not** eligible to be copied. Note that, although not eligible for copy, shadow lines representing NOTSELECTED or SUPPRESSED records groups are still included within a C(n) count of lines to be copied. If a shadow line is not displayed (SHADOW OFF), it is not included within a C(n) line count.
D(n) DD	Delete a line or a block of lines. Alternatively, for RRDS/VRDS Update-in-place editing, selects lines to become Empty Slots. Only visible (not excluded) data records and shadow lines representing EXCLUDED record groups are eligible for delete. i.e. records flagged as NOTSELECTED, SUPPRESSED or, if SHADOW OFF EXCLUDED is in effect, records flagged as EXCLUDED, are **not** eligible to be deleted. Note that, although not eligible for delete, shadow lines representing NOTSELECTED or SUPPRESSED records groups are still included within a D(n) count of lines to be deleted. If a shadow line is not displayed (SHADOW OFF), it is not included within a D(n) line count.
F(n)	Show the first n records of an excluded record group.
FMT	Display the record or record segment in single record, formatted view. Prefix commands FMT and MAP are synonyms.
HEX or HEXD	Display the record or record segment in HEXDUMP format.
I(n)	Insert a new line or a block of n new lines of the default record type. If display format is TABLE, then numeric/bit fields are initialised to zero and character fields to blank, otherwise the whole record is initialised to blank. For RRDS/VRDS Update-in-place editing, I(n) selects a line or block of lines to be searched for empty slots. All empty slots found within this range of lines will be activated. Shadow lines representing NOTSELECTED or SUPPRESSED records groups are still included within an I(n) count of lines to searched. If a shadow line is not displayed (SHADOW OFF), it is not included within an I(n) line count.
ID(n) IDD	Remap (IDENTIFY) a line or a block of lines. IDENTIFY will action SELCOPY/i SDE record type assignment processing to re-assign a new record type to the records/segments in the selected lines.
L(n)	Show the last n records of an excluded record group.
M(n) MM	Mark a line or a block of lines for move. Lines may be moved to the clipboard (effectively performing a CLIPBOARD CUT operation) or moved to another position within the same edited data using prefix commands, A or B. Only visible (not excluded) data records and shadow lines representing EXCLUDED record groups are eligible to be moved. i.e. records flagged as NOTSELECTED, SUPPRESSED or, if SHADOW OFF EXCLUDED is in effect, records flagged as EXCLUDED, are **not** eligible to be moved. Note that, although not eligible for move, shadow lines representing NOTSELECTED or SUPPRESSED records groups are still included within a M(n) count of lines to be moved. If a shadow line is not displayed (SHADOW OFF), it is not included within a M(n) line count.
MAP	Display the record or record segment in single record, formatted view. Prefix commands FMT and MAP are synonyms.

R(n) RR(n) "(n) ""(n)	Replicate (duplicate) a line or a block of lines n times. Only visible (not excluded) data records and shadow lines representing EXCLUDED record groups are eligible for replication. i.e. records flagged as NOTSELECTED, SUPPRESSED or, if SHADOW OFF EXCLUDED is in effect, records flagged as EXCLUDED, are **not** eligible to be replicated.
STP	Applicable to edit of Segmented Records only. Force a secondary segment to be a primary (base) segment so splitting the record into two. No action is taken if the segment is already a primary segment type. An IDENTIFY operation is automatically performed on the focus record to remap all segments with appropriate record types.
STS	Applicable to edit of Segmented Records only. Force a primary segment to be a secondary segment so joining the record with the record before. No action is taken if the segment is already a secondary segment type. An IDENTIFY operation is automatically performed on the focus record to remap all segments with appropriate record types.
V	Display only records that are of the same record type as this line. V may be specified on a visible data record or on EXCLUDED, SUPPRESSED or NOTSELECTED shadow lines. The DRECTYPE value is updated to be this record type and this line becomes the first line of the display.
V+	Add to the display records that are of the same record type as this line. V+ may be specified on a visible data record or on EXCLUDED, SUPPRESSED or NOTSELECTED shadow lines. Note, however, that specification on a visible data record or EXCLUDED record group shadow line will not add new record types to the display. The DRECTYPE value is updated to be this record type and this line becomes the first line of the display.
V-	Remove from the display records that are of the same record type as this line. V- may be specified on a visible data record or on EXCLUDED, SUPPRESSED or NOTSELECTED shadow lines. Note, however, that specification on a SUPPRESSED or NOTSELECTED record group shadow line will not remove record types from the display. The DRECTYPE value is updated to be this record type and this line becomes the first line of the display.
X(n) XX	Mark a line or a block of lines for exclusion from the display. Only visible (not excluded) data records are eligible to be excluded. i.e. records flagged as NOTSELECTED, SUPPRESSED or EXCLUDED are **not** eligible to be excluded. Note that, although not eligible for exclusion, shadow lines representing NOTSELECTED, SUPPRESSED or EXCLUDED records groups are still included within an X(n) count of lines to be excluded. If a shadow line is not displayed (SHADOW OFF), it is not included within an X(n) line count.
Z	Switch to a zoomed (single record view) display of the record occupying this line.

Function Keys

3270 Program Function Keys (PFKeys) may be assigned to SDE commands.

The default SDE function keys may be tailored in the **(SDE)** section of the System and User INI files.

Following startup of SELCOPY/i, individual SDE PFKeys may be assigned new definitions using the SELCOPY/i KEYS command.

The **KEYS** command may also be used to open the Function Keys window to display, and optionally update, the current function key settings.

The SDE program default function keys are:

PF1	insert	Insert a new record following the focus line.
PF2	macro SdeZoomW	Open a new window in ZOOMed-mode.
PF3	end	Quit the file (you will be prompted to save any changes).
PF4	macro SdeUtil	Open the SDE Edit/Browse Utilities Menu.
PF5	rfind	Locate search string defined by last FIND or CHANGE command.
PF6	rchange	Repeat the change requested by the last CHANGE command.
PF7	up	Scroll the window display upwards by an amount determined by the scroll field or specified on the command line.
PF8	down	Scroll the window display downwards by an amount determined by the scroll field or specified on the command line.
PF9	MDINext	Place focus on the next MDI child window. (Edit view or list.)
PF10	left	Scroll the window display to the left by an amount determined by the scroll field or specified on the command line.
PF11	right	Scroll the window display to the right by an amount determined by the scroll field or specified on the command line.
PF12	retrieve -	Retrieve the last command to the command line.
PF13	delete	Delete the focus line.
PF14	macro SdeSel	Create a temporary CMX file containing an SDE SELECT command so allowing the user to choose the field columns to be displayed.
PF15		
PF16		
PF17		
PF18		
PF19		
PF20		
PF21		
PF22		
PF23		
PF24		

Note that, if the following conditions are true, the contents of the command line is concatenated to the definition of the function key and the result executed as a single command.

1. The function key command is **not** set to execute before processing any user screen inputs.
   This is determined by the status of the **BEFORE** field for the PFKey definition. (See the Function Keys window opened from an SDE Edit/Browse view.)

2. The concatenation of the PFKey function and the command line contents does not result in an invalid ISPF scroll command (UP, DOWN, LEFT, RIGHT.)

   Where an invalid scroll command is the result then, if the first parameter on the scroll command begins with a numeric character, an error is returned. Otherwise the contents of the command line are executed before the PFKey function.

Glossary

The following is a glossary of terms used in this document.

CLI (**Command Line Interface**)
A Command Line Interface is a text based method by which users can execute functions supported by the application.

CBLe
A powerful text editor that runs as an MDI application under SELCOPY/i. CBLe supports its own CLI and has been developed based on specifications found in Mansfield Software's KEDIT for Windows.

Current Column
The first field column visible within the current display area, belonging to records of the default record type. The current column references the same field within all data records that are of the default record type.

Current Line
The first record group (data record or shadow line) displayed within the current display area view.

Current Record Group
The record group occupying the current line.

Current SDE Window
The SDE Edit View which received focus last.
The current SDE window prevails even if the current focus window is not an SDE view. The concept of a current SDE window is important when executing CBLe/SDE editor REXX macros or when executing SDE commands from a CBLe edit view via the SDATA command.

Default Record
The focus line if it contains a visible data record, otherwise the first visible data record following the focus line that is of the default record type.

Default Record Type
The default record type is defined as being the record type of the focus line if the focus line is a visible or EXCLUDED record group shadow line, otherwise it is the record type defined by DRECTYPE setting.
See section *Default Record Type*.

Focus Column
The field column on which the cursor is positioned within the focus line.
If the focus line is **not** a visible or EXCLUDED record group, or if the cursor is positioned outside the display area (e.g. the command line) or within the prefix area, the focus column is defined as being the current column. The focus column references the same field within all data records that are of the same record type as the focus line.

Focus Field
The individual field within the focus line referenced by the focus column.

Focus Line
The line within the display area on which the cursor is positioned.
If the cursor is positioned on a header line, the focus line is the first line that immediately follows the header line. If the cursor is positioned outside the display area (e.g. the command line), then the focus line is defined as being the current line.

Focus Record Group
The record group occupying the focus line.

Header Line
A line within the SDE window display area that contains column header information for the record group that follows. A header line constitutes one line within a group of header lines which scroll with the record data display.

INI file
File containing configuration options for SELCOPY/i. The System INI file is processed on startup of SELCOPY/i and contains options that apply to all users. The User INI file contains options specific to each user that may, where appropriate, override options set in the System INI file.

List Window
A SELCOPY/i window containing rows of associated information. List windows support point-and-shoot column sorting; select, sort and filter CLI commands; and prefix area commands.

MDI
Multiple Document Interface is a Microsoft specification for PC applications that enable the user to work with multiple documents at the same time. Each document is displayed in a separate child window within the client area of the application's main (frame) window. Typical MDI applications on PCs include word-processing and spread sheet applications.

MDI Client Area window
The MDI client area window is the display area within an MDI application's frame window. The MDI client area serves as the backround for MDI child windows.

MDI Child/Document Window
An MDI child or document window is opened in an application's client area window each time a document is opened. Each child window has a sizing border, title bar, window menu, minimise, maximise, restore and close buttons. A child window is clipped so that it is confined to the client window and cannot appear outside it.

When a child window is maximized, its client area completely fills the MDI client area window. In addition, the system automatically hides the child window's title bar, and adds the child window's window menu icon and Restore button to the MDI application's menu bar.

MDI Frame Window
An MDI frame window may be considered the main window of an MDI application. It is the parent window of the MDI client area window in which MDI child windows are opened. It has a sizing border, title bar, window menu, minimise, maximise restore and close buttons.

Record Group
A record group is one or more data records that is represented by a single line in the SDE window display.
A visible data record may be considered to be a record group of one record wheras a shadow line may be a record group of one or more consecutive records of the same record type.

Record Type
Term referring to the name assigned to a record type object (RTO) on execution of the CREATE STRUCTURE command.
This name is the highest level field name identifier for a record mapping in a COBOL or PL1 copybook or in a SELCOPY/i SDE data definition clause.

RTO (**Record Type Object**)
A single record type definition within an SDO.
RTO definitions are generated via the CREATE STRUCTURE command which uses record structures defined with SELCOPY/i's own SDE structure definition syntax or defined from within an existing COBOL or PL/1 copybook.
An RTO (and hence the SDO) may subsequently be altered (e.g. via the USE command) and saved to an SDF.

SD (**Structured Data**)
Structured Data refers to data within file records that have a pre-defined structure. See structured records.
SD is also the minimum abbreviation for SDATA, the CBLe CLI command used to prefix any of SELCOPY/i's SDE CLI commands when executed from within a CBLe edit view.

SDF (**Structure Definition File**)
A disk file (sequential data set or PDS/PDSE member) containing a saved Structure Definition Object (SDO).
If not already in storage, an SDF gets loaded (creating an SDO) during execution of an EDIT or BROWSE command in order to apply a structure to records processed by SELCOPY/i's SDE.
If an SDO is altered during an SDE edit session and is not flagged as being temporary, then the user will be prompted to save the SDO to an SDF. An SDO may also be saved to an SDF automatically during a CREATE STRUCTURE command.

SDE (**Structured Data Environment**)
Structured Data Environment that runs under SELCOPY/i and includes MDI display windows, processing options and CLI commands. SDE enables users to browse, edit, update, copy and compare data in records that are mapped by a SELCOPY/i defined structure or COBOL, PL1 copybook.

SDE Edit View
An SDE MDI document window that contains a display of structured data. If the same file is displayed in multiple windows, then the user has multiple SDE edit (or CBLe text edit) views of the file.

SDO (**Structure Definition Object**)
An in-storage structure definition consisting of one or more record type objects (RTO) that map records in a structured file.
An SDO is created by a CREATE STRUCTURE command or an EDIT or BROWSE command if the SDF is not already loaded in storage.

SELCOPY/i
The Interactive environment developed by CBL and supplied as part of SELCOPY and CBLVCAT licensable software products.

Structured Data Set
A data set containing structured records.

Structured Records
Records that consist of one or more data fields, each with a defined field start position, length and data type.